Kingston Custom Business Applications

Last updated: June 22, 2020
Some examples and graphics depicted herein are provided for illustration only. No real association or connection to ServiceNow products or services is intended or should be inferred.

If you have comments about this documentation, submit your feedback to:
docfeedback@servicenow.com
# Table of Contents

**Now Platform Custom Business Applications**

- Applications .................................................. 4
- Parts of an application ..................................... 4
- Contextual development environment ............... 20

**Creating applications**

- Video: creating an application ............................ 57
- Basic development process ................................. 58
- Delegated development ...................................... 71

**Application management**

- Install an application ........................................ 77
- Install an update ................................................ 78
- Select an application from the applications list ...... 78
- Select an application from the application picker ... 78
- Preserve unpublished applications during a system clone . 79
- Application sharing ........................................... 81
- Application removal .......................................... 87
- Move an application file between global applications . 89

**Application tools**

- Automated Test Framework ............................... 90
- ServiceNow Studio .......................................... 239
- Service Creator ............................................. 266
- Team Development ........................................... 280
- ServiceNow application repository ...................... 324
- System update sets .......................................... 327

**APIs and scripts**

- JavaScript API reference .................................... 375
- REST API reference .......................................... 1594
- Scripts .......................................................... 1810

**Web services**

- REST API ...................................................... 2078
- REST API reference .......................................... 2079
- SOAP web service .......................................... 2102
- Scripted REST APIs .......................................... 2136
- Inbound web service examples ......................... 2238
- Excel web service ........................................... 2263
- CSV Web service ............................................ 2264
- JSONv2 Web Service .......................................... 2265
- PDF web service ............................................. 2275
- XML web service ............................................ 2276
- RSS web service ............................................. 2279
- Web services security ....................................... 2286
- REST & SOAP API analytics ............................... 2286
- Outbound REST web service .............................. 2291
- Outbound SOAP web service ............................. 2308
- Outbound web services mutual authentication ....... 2325
- Outbound web services logging ......................... 2326
- Server Name Indication for outbound web services .. 2330
- HTTP Client Connection Management .................. 2331
- ODBC driver .................................................. 2332
- Domain separation in Web Services ................. 2385

**Index** .......................................................... 2386
Now Platform Custom Business Applications

ServiceNow provides a single mobile and web application development platform to quickly build business applications that power your digital transformation.

Applications

Applications are a collection of files and data that deliver a service and manage business processes.

By default, the Now Platform offers a suite of standard applications such as Incident, Problem, and Change, to provide service management and service automation. If you have additional business needs, you can also:

- Install applications
  - Install from an application repository
  - Install from the ServiceNow Store
- Update installed applications
- Create applications
  - Publish to an application repository
  - Publish to the ServiceNow Store

Parts of an application

Applications consist of several types of files and records that collectively deliver a service.
Custom application record

The custom application record defines and identifies an application and all its associated artifacts.

It is similar to a system dictionary record for a table or column in that it stores the most current configuration of an application. The system automatically creates a custom application record during the application creation process. Application developers can use this record to perform the following tasks.

- Change the application name
- Change the application version
- View the scope the system uses to identify application files and configuration records
- Enable scoped administration
- Manage design and runtime access to the application
  - Select what JavaScript standards the application supports
  - Select how the system tracks runtime API operations
  - Permit or restrict access to tables from other applications
- Monitor or enforce subscriptions
- Select the default menu in which to display application modules
- Set the user role required to access the application
- Add or update a logo
- View all application files
- View resources from other applications on which the application depends
- View the run-time resource to which the application has been granted access
- View the design-time resources to which the application has been granted access

Application versions

Each installed application has a version as defined by its application developer in the custom application record.

The system uses this version information to determine whether there are updates available from the ServiceNow application repository or ServiceNow Store.

Application scope

Application scoping protects applications by identifying and restricting access to application files and data.

Administrators can specify what parts of an application are accessible to other applications from:

- The Custom application record
- Each application Table record

For example, suppose that you create a conference room booking application in its own application scope. By default, the application can access and change its own tables and business logic but other applications can't unless you give them explicit permission. The application scope ensures:

- The conference room booking application does not interrupt core business services.
- Other applications do not interfere with its normal functioning.
By default, all custom applications have a private scope that uniquely identifies them and their associated artifacts with a namespace identifier. The application scope prevents naming conflicts and allows the contextual development environment to determine what changes, if any, are permitted. Application developers specify an application scope when they create an application.

**Tip:** Global apps can alter data that you don’t intend to alter. You should leverage scoped apps to create new tables, and consider transitioning old ones to scoped apps. This allows you to split responsibilities with Delegated Development. To learn more about delegated development, see [Delegated development](#).

**Global scope**

The global scope is a special application scope that identifies applications developed prior to application scoping or applications intended to be accessible to all other global applications.

Applications in the global scope do not append a unique namespace identifier to the application name. Global applications can have naming conflicts and data collisions when developers create multiple global applications with the same name.

Since all global applications are in the same application, they bypass scope protections. Global applications allow other global applications access to their tables to

- Read records
- Run API requests
- Create configuration records

Typically, only applications provided by ServiceNow are in the global scope, however any custom applications created before application scope was implemented are also in the global scope.

Applications in the global scope are also not eligible for upload to the application repository or the ServiceNow Store.

**Namespace identifier**

The system adds a namespace identifier to the front of application artifacts such as tables, scripts, and configuration records.

The identifier cannot be changed or removed from application artifacts to ensure that they are always associated to the proper application and that they have a unique name.

The system generates a namespace identifier from the following information:

<table>
<thead>
<tr>
<th>Elements used to generate a namespace identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>The prefix characters for a scoped application.</td>
</tr>
<tr>
<td>The instance customer prefix (glide.appcreator.company.code)</td>
</tr>
</tbody>
</table>
The example values generate a namespace identifier of \textit{x\_acme\_book\_rooms}.

\textbf{Namespace identifier examples}

The following examples illustrate generating namespace identifiers for applications, tables, and fields.

\textbf{Example namespace identifiers}

<table>
<thead>
<tr>
<th>Action</th>
<th>Element generated</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Generate a namespace identifier for a private scope application called Book Rooms.</td>
<td>x_acme_book_rooms</td>
<td>This is a combination of the vendor prefix and application ID.</td>
</tr>
<tr>
<td>2. Generate a namespace identifier for a global scope application called Marketing Events.</td>
<td>None</td>
<td>The system does not generate namespace prefixes for global applications.</td>
</tr>
<tr>
<td>3. Add the conference rooms table to the Book Rooms application.</td>
<td>x_acme_book_rooms_conference_rooms</td>
<td>This table is in the Book Rooms scope so begins with the namespace identifier.</td>
</tr>
<tr>
<td>4. Add a Marketing Event table to a global application.</td>
<td>u_marketing_event</td>
<td>Custom tables in the global scope always use the u_ namespace identifier.</td>
</tr>
<tr>
<td>5. Select Book Rooms in the application picker and add the \textbf{Capacity} field on the Conference Rooms table.</td>
<td>capacity</td>
<td>The field is in the same scope as the table so it does not have its own namespace identifier. However, to dot-walk to the field in a script, you would use the full path including the table namespace identifier: x_acme_book_rooms_conference_rooms.capacity.</td>
</tr>
<tr>
<td>6. Select Book Rooms in the application picker and add the \textbf{Theme} field to the Marketing Event table.</td>
<td>x_acme_book_rooms_theme</td>
<td>The field is in a different scope from the table so it is prefixed with the x_acme_book_rooms namespace identifier. To dot-walk to the field in a script, you would use full path including the field namespace identifier: u_marketing_event.x_acme_book_rooms.theme.</td>
</tr>
</tbody>
</table>

\textit{Note: } This example assumes that the Marketing Event table allows other application scopes to add fields. For more information, see Application Access Settings.
Application tables

Application developers create tables and their associated lists and forms for users to add and update records.

An application owns its tables and determines whether other applications can access resources from them. For example, the Book Rooms application can store conference room data in the Conference Rooms (x_acme_book_rooms_conference_rooms) table and permit other applications to read this data.

The system uses standard access controls to manage user access to application data. During application creation, developers can specify an application-specific user role for these access controls. They can also use application access settings to manage runtime and design time access to application tables.

User interface elements

By default, the system creates a list and form view for each table.

Application developers can configure the layout of these views to provide a basic user interface. In addition, they can also create supporting menus, modules, or UI pages to access these list and form views.

Alternatively, developers can create their own custom user interface with Service Portal or the Content Management System.

Application user roles

Application developers can create application-specific user roles to control access to application data.

Application developers can create an end-user role for the primary users of a custom application. The system automatically restricts access to application data by:

- Standard record operations on application tables are restricted to the application user role or the admin role.
  - Create
  - Read
  - Write
  - Delete
- The application menu is restricted to the application user role or the admin role.
- All application modules are restricted to the application user role or the admin role.

Application developers can create additional user roles to support the application but must manually create any associated access controls or role requirements.

Web services integrations

Application developers can create web services integrations to application tables.
By default, the system allows other applications to access application tables using web services integrations. Application developers can allow or restrict web services access from the table application access settings.

**Dependencies for custom applications**

Every custom application record includes a related list identifying its dependencies on other applications.

Administrators can review this list to determine whether an application poses any risk to existing processes or data. Application developers can use this list to ensure that their applications have the proper access to other applications.

---

**Sample application dependencies**

**Application files**

Application files are configuration records that allow developers to extend application functionality.

Application developers create application files when they add configuration records for application logic such as business rules, workflows, and script includes. Scoped applications do not own these tables, but they do own the records (files) within these tables. For example, adding a business rule to check for available rooms from the Conference Room table adds an application file to the Business Rule (sys_script) table. Application developers can view the complete list of application files from the custom application record.

The Application File (sys_metadata) table is the parent table for all tables that contain configuration records. It provides a series of standard fields that define the attributes for a
configuration record. Tables that contain configuration records extend the Application File table. For example, the Business Rule (sys_script) table extends the Application File table.

Developers do not create application file records directly from the Application File table. Instead, they create or modify configuration records, and the system creates or modifies the associated application file record. Most configuration record tables do not display a reference field or related list for the Application File table. By default, only Applications (sys_scope) and Tables (sys_db_object) have a reference to the Application File table.

The system also tracks application file records in Update Sets. Whenever you change an application file record or its related configuration record, the system adds a corresponding record in the Customer Updates (sys_update_xml) table. The system uses a single update record, ensuring that a configuration record and its application file are kept in sync when transferring applications between instances.

Administrators can:

- View file properties for configuration records.
- Protect application files from changes during upgrades.
- View parent-child relationships between configuration records.

View file properties

Administrators can view the application file properties of a single record.

1. Navigate to the form view of the configuration record. For example, navigate to System Definition > Business Rules and select a business rule for the Incident table.
2. Right-click the form header and select Show File Properties.
The Application File table provides the standard fields that define the attributes for the configuration record.

3. To return to the configuration record view, click the **Show Related Record** related link.
<table>
<thead>
<tr>
<th>Display name</th>
<th>Caller Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Global</td>
</tr>
<tr>
<td>Update name</td>
<td>sys_script_28beab035f201000b12e3572f</td>
</tr>
<tr>
<td>Protection policy</td>
<td>-- None --</td>
</tr>
<tr>
<td>Class</td>
<td>Business Rule</td>
</tr>
<tr>
<td>Customer update</td>
<td></td>
</tr>
</tbody>
</table>

### Accessed by

**Created**

2011-08-22 10:46:53

**Updated**

2011-08-22 11:42:14

**Created by**

admin

**Updated by**

admin

### Related Links

- Show Related Record
- Show Parent Record
4. Navigate between a customer update record, the file properties view, and the configuration record view.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Related Record related link</td>
<td>Navigate to the configuration record</td>
</tr>
<tr>
<td>Show Parent Record related link</td>
<td>Navigate to the parent record of the current configuration record.</td>
</tr>
<tr>
<td>Descendants related link</td>
<td>View child configuration records, such as a field label translation.</td>
</tr>
</tbody>
</table>

**Application File form**

Use the Application File form to view relationships between applications and configuration records.

**Application File form fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display name</td>
<td>Display name for the configuration record.</td>
</tr>
<tr>
<td>Update name</td>
<td>Unique identifier for the configuration record. This value is used to identify versions and updates of the record.</td>
</tr>
<tr>
<td>Class</td>
<td>Table that contains the configuration record.</td>
</tr>
<tr>
<td>Application</td>
<td>Application that contains the configuration record.</td>
</tr>
<tr>
<td>Protection policy</td>
<td>Policy that determines if the configuration record is protected from changes. See Protected Application Files.</td>
</tr>
<tr>
<td>Created</td>
<td>Creation date of the configuration record.</td>
</tr>
<tr>
<td>Created by</td>
<td>User who created the configuration record.</td>
</tr>
<tr>
<td>Updated</td>
<td>Last update date for the configuration record.</td>
</tr>
<tr>
<td>Updated by</td>
<td>User who last updated the configuration record.</td>
</tr>
<tr>
<td>Related Record Versions</td>
<td>Version records for the related configuration record. A version record is created every time a user changes the related record. Use this list to compare versions of the configuration record or to revert to a previous version. See Versions.</td>
</tr>
<tr>
<td>Related Record Updates</td>
<td>Local update records for the related configuration record. An update record is created for the most recent change to the related record in a given Update Set. See Update Sets.</td>
</tr>
</tbody>
</table>

**Note:** You may find it useful to manually add an **Owned By** (owned_by) field to this form. It indicates, for each application, who in IT owns the application and is responsible for maintaining information about it. The Data Certification function uses this field to send certification tasks every quarter, or at any time interval that is configured. You can assign certification tasks to other users, but using this field limits configuration effort. To learn more
about adding a field to a form, and about Data Certification, see Using the form designer and Data Certification

Application file protection policy

A read-only protection policy prevents anyone from modifying an application file or its related record.

Some application code shipped with the ServiceNow system requires special protection. Only a ServiceNow employee can alter the protection policy and then modify the application file or its related record. A ServiceNow employee cannot edit protected files without changing the policy designation first.

To prevent customizations from being overwritten by system upgrades, the upgrade process automatically skips changes to customer-updated records. If you modify an application file or related record that is later designated as Read-only in an upgrade, the application file maintains the default protection policy of Write. You keep the existing modifications and can continue modifying the records.

Note: Reverting a customized file to its baseline state causes the record to inherit the new protection policy as well. For example, a record going from a Write protection policy to a Read-only protection policy.

Relationships between configuration records

The Application File Types table defines parent-child relationships between configuration records.

The system uses this structure to keep configuration records that normally belong together in the same application.

Note: Do not modify the Application File Types table as it provides system functionality.

For example, consider the parent-child relationships for a UI policy:

- The UI policy is a child of the application table.
- UI policy actions are children of the UI policy.
- UI policy actions have a parent UI policy and a grandparent application table.
- The UI policy actions and the UI policy are all descendants of the application table.
Fix scripts

A fix script is server-side JavaScript code that you run after an application is installed or upgraded. Include fix scripts to make changes that are necessary for the data integrity or product stability of an application.

Administrators and users with the script_fix_admin role can create and run fix scripts.

Create a fix script

Create fix scripts to ensure the system installs or updates an application properly.

Use fix scripts to add, update, and delete data, including rules, scripts, and property settings.

1. Navigate to System Definition > Fix Scripts.
2. Click **New**.
3. Define the fix script by completing the fields on the form.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a unique, descriptive name for the fix script.</td>
</tr>
<tr>
<td>Active</td>
<td>Select the check box to enable the fix script.</td>
</tr>
<tr>
<td>Unloadable</td>
<td>Select the check box to create Customer Update (sys_update_xml) records when the fix script runs. Clear the check box to run upgrades without creating these records (default). Unloadable is enforced when you test the fix script.</td>
</tr>
</tbody>
</table>
| Run once     | Select the check box to run the script only one time (default). The fix script is ignored for subsequent upgrades. Clear the check box allow the script to run every time the application is installed or upgraded. This option is enforced by the following mechanism.  
- When a fix script is processed, a corresponding record is added to the sys_update table.  
- The upgrade process automatically skips fix scripts that are flagged as Run once and have an entry in the sys_update table. Run once is not enforced when you test the fix script. |
| Flush cache  | Select the check box to require a cache flush after the application is installed or upgraded. Certain system changes—such as changes to the dictionary, client scripts, UI policy, or system properties—require a cache flush to take effect. If the check box is selected for at least one of the fix scripts that are run, the cache is flushed when the installation or upgrade is complete. If the check box is not selected for any of the fix scripts that are run, then the cache is not flushed. Flush cache is enforced when you test the fix script. |
| Before       | Select the check box to run the fix script before installing or upgrading the application. Clear the check box to run the fix script after (default). |
| Description  | Enter a description of the fix script.                                                                                                 |
| Script       | Enter the code for the fix script.                                                                                                        |
4. Click **Submit**.
5. Test the fix script and make any necessary changes.

**Test a fix script**

Test your fix scripts to ensure they install or update applications as expected.

*Fix scripts add, update, and delete data, including rules, scripts, and property settings.*

**Note:** Do not test fix scripts on production instances.

1. Open the fix script record.
2. Review the code design to ensure that it can run more than once on the same system without causing damage.

   This review is different from the Run once option, which determines whether it is necessary to run it more than once on the system. For example, you may write a fix script that adds a role to a property by default. Design the script so that it can run multiple times on the same system without overwriting the existing data, even if it is not necessary to run the script again after the initial installation.

3. Click the **Run Fix Script** related link.

   The **Run once** option is not enforced for tests, however the **Flush cache** and **Unloadable** options are enforced.

4. Confirm how to run the script.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proceed in Background</strong></td>
<td>Use this option for long-running scripts, or if you do not know the expected execution time.</td>
</tr>
<tr>
<td><strong>Proceed</strong></td>
<td>Use this option to run the script immediately and display the results in a confirmation window.</td>
</tr>
</tbody>
</table>

5. Review the results from the Progress Workers related list, and make any necessary changes. To manually stop a running fix script:
   a) From the **Progress Workers** related list, select a worker in the **Running** state.
   b) Select the **Cancel job** related link.
Run fix scripts

After you transfer an application to another instance, you must manually run any necessary fix scripts.

To run a fix script:

1. Navigate to System Definition > Fix Scripts.
2. Edit the filter to search for your application name. For example, Application | is | Book Rooms).
3. Open the fix script record.
4. Click Run Fix Script.

Fulfillment tables

To enable a production instance to enforce entitled usage of your ServiceNow Store App, you configure the tables where only record owners or subscribed app users can make updates.

For any table that you, the developer, create or extend to support a custom application, you can specify that the table is a fulfillment table. In a fulfillment table, only a subscribed fulfiller user can perform a fulfiller action (typically, create/update/delete a not-own record).

In contrast, for a table that is not a fulfillment table, any user—even a user who is not subscribed—can act as a requester. The intent is to allow the usage admin to enable subscription enforcement on any production instance that implements the application.

Ownership of records in a fulfillment table

To enable the system to identify a fulfiller action, you define how to determine ownership of any record in the table. The developer of the application specifies the set of conditions that determine whether a user owns the record. For example, UserA owns a record in a task table if UserA opened the record or another resource opened the record on behalf of UserA.

For task-extended tables, time cards, and apps that require a subscription, the system sets the table as a fulfillment table by default and auto-assigns the ownership condition. For tables that you create to support your app, you can mark the table as a fulfillment table and can specify the ownership condition (for example, use the filter (opened_by)(is)(currentUser) OR (caller_id)(is)(currentUser)).
System default conditions for ownership

<table>
<thead>
<tr>
<th>Action</th>
<th>Ownership condition (owner_condition)</th>
</tr>
</thead>
<tbody>
<tr>
<td>task extension</td>
<td>opened_by (read-only)</td>
</tr>
<tr>
<td>catalog request</td>
<td>requested_for (read-only)</td>
</tr>
<tr>
<td>other tables in apps that require a subscription</td>
<td>sys_created_by (read-only)</td>
</tr>
<tr>
<td>tables created by developer for app that requires a subscription</td>
<td>Specified by developer</td>
</tr>
</tbody>
</table>

Specify that a table is a fulfillment table

You configure a table as a fulfillment table to enable the system to prevent updates by users who are not subscribed to the app. For ServiceNow Store apps, you configure a table as a fulfillment table to enforce that fulfillment usage complies with your subscription use policy.

Role required: usage_admin, admin

1. While working on a table, open the Subscription Management related list.
2. Select Fulfillment table.
3. Specify how the system determines ownership of records in the table so that both end users who own a record and subscribed fulfiller users can update the record: Specify the Ownership condition. For example, set the filter as (opened_by) (is) (currentUser) OR (caller_id) (is) (currentUser).

Contextual development environment

The platform is a contextual development environment that displays the currently selected application, identifies the scope of every application artifact, and prevents any changes that violate the access settings for an application.

Application developers can use the contextual development environment to:

- Determine the application context
- View and select applications
- Enforce application version standards
- Enforce application resource throttling
- Enforce script protection policies

Application context

When application developers create new records, the system automatically assigns the records to the currently selected application in the application picker.

When application developers attempt to change existing records, the platform checks whether the currently selected application matches the scope of the application artifact. If they match, the application developer can save changes to the artifact. If they differ, the system makes the following changes to the user interface:

- Makes all the fields on the current record read-only.
- Displays a warning message that the application artifact belongs to another scope.
Application access settings

Application access settings determine whether one application can access resources from another application.

Application access settings are similar to access controls (ACLs) in that they allow you to restrict access to certain resources, but instead of restricting tables and records from users they restrict applications resources from other applications. There are several ways to set cross-scope access.

<table>
<thead>
<tr>
<th>Setting type</th>
<th>Use</th>
<th>Description</th>
<th>Access setting location</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application design and runtime settings</strong></td>
<td>Your application or application script requires access to a cross-scope resource.</td>
<td>Application designers can configure the following application access settings for the entire application.</td>
<td>Access determined by the cross-scope privilege record owned by the calling application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Specify whether cross-scope tables can be selected during design-time activities.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Specify when scripts can run on cross-scope resources.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Specify what JavaScript standard the application supports.</td>
<td></td>
</tr>
<tr>
<td><strong>Table design and runtime settings</strong></td>
<td>A cross-scope application or web service requires access to perform CRUD operations on a table.</td>
<td>Application developers can also configure application access settings for individual tables.</td>
<td>Access determined by settings on the target table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Specify whether the table is available to other application scopes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Specify what runtime operations from other application scopes the table supports.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Specify whether the table can be selected during design-time activities.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Specify whether the table can be accessed from web services.</td>
<td></td>
</tr>
<tr>
<td><strong>Application restricted caller access settings</strong></td>
<td>A cross-scope application or script requires access to your application or application resource.</td>
<td>Admin users can configure the following application access settings for an entire application scope or application resource.</td>
<td>Access determined by the restricted caller access record owned by the target application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Specify whether a cross-scope script can access your application scope or an application resource.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Track cross-scope requests for access to application resources.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Approve or deny cross-scope requests to access application resources.</td>
<td></td>
</tr>
</tbody>
</table>

Application design and runtime settings

The application design and runtime settings determine whether an application can access cross-scope resources.
Design and Runtime fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Mode</td>
<td>The application's supported JavaScript standard. Select <strong>ES5 Standards Mode</strong> to support features in ECMAScript 5th edition. Select <strong>Compatibility Mode</strong> to support earlier ECMAScript editions.</td>
</tr>
<tr>
<td>Runtime Access Tracking</td>
<td>The application's handling of script access requests to resources in other applications. Select <strong>None</strong> to authorize all access requests to cross-scope resources without logging them. Select <strong>Tracking</strong> to log and authorize all access requests to cross-scope resources. Select <strong>Enforcing</strong> to log access requests to cross-scope resources but require an administrator to authorize each request.</td>
</tr>
<tr>
<td>Restrict Table Choices</td>
<td>The availability of cross-scope tables when designing the application. Clear the option to allow the application to see tables from other application scopes. Select the option to restrict design choices to only tables in the same application.</td>
</tr>
</tbody>
</table>

Runtime access tracking

Runtime access tracking allows administrators to manage script access to application resources by creating a list of script operations and targets that the system authorizes to run.

Runtime access tracking provides the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Tracking done</th>
<th>Authorization done</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>The system does not track runtime access requests.</td>
<td>The system does not require authorization to run access requests.</td>
<td>Application scripts can access resources from other applications as long as the table-level access settings allow it.</td>
</tr>
<tr>
<td>Tracking</td>
<td>During development, the system creates a Cross-Scope Privilege record for each runtime access request. After installation, the system no longer tracks new runtime access requests.</td>
<td>The system sets the status of the Cross-Scope Privilege record to <strong>Allowed.</strong></td>
<td>The system runs the tracked operation as long as the table-level access settings allow it.</td>
</tr>
<tr>
<td>Option</td>
<td>Tracking done</td>
<td>Authorization done</td>
<td>Results</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>--------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Enforcing</td>
<td>During development, the system creates a Cross-Scope Privilege record for each runtime access request. After installation, the system no longer tracks new runtime access requests.</td>
<td>The system sets the status of the Cross-Scope Privilege record to <strong>Requested</strong>.</td>
<td>The system blocks the tracked operation from running until an Administrator manually changes the status to <strong>Allowed</strong> and the table-level access settings allow it.</td>
</tr>
</tbody>
</table>

During development, application designers must run all of an application's script logic to ensure the system tracks and authorizes the access requests to other applications.

**Cross-scope privilege record**
Runtime access tracking uses cross-scope privilege records to determine which script operations and targets the system allows to run.

The system creates cross-scope privilege records when:

- Runtime access tracking is set to **Tracking** or **Enforcing**.
- A script attempts to access another application.

Each cross-scope privilege record contains the following information.

**Cross-scope privilege fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Scope</td>
<td>The application requesting runtime access to another application's resources.</td>
</tr>
<tr>
<td>Target Scope</td>
<td>The application whose resources are being requested.</td>
</tr>
<tr>
<td>Target Name</td>
<td>The name of the table, script include, or script object being requested.</td>
</tr>
<tr>
<td>Target Type</td>
<td>The type of request: table, script include, or script object.</td>
</tr>
<tr>
<td>Operation</td>
<td>The operation the script performs on the target. The target type determines the available operations. Tables support the read, write, create, and delete operations. Script includes and script objects only support the execute API operation.</td>
</tr>
<tr>
<td>Status</td>
<td>The authorization for this record: requested, allowed, or denied</td>
</tr>
</tbody>
</table>

Administrators can manually create cross-scope privilege records for application developers in advance to communicate which cross-scope resources they expect developers to access. For example, administrators could create these cross-scope privilege records to permit application developers access to resources from Incident Management.

**Sample cross-scope privilege records**

<table>
<thead>
<tr>
<th>Source Scope</th>
<th>Target Scope</th>
<th>Target Name</th>
<th>Operation</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>My App</td>
<td>Global</td>
<td>incident</td>
<td>Read</td>
<td>Allowed</td>
</tr>
</tbody>
</table>
During testing, application developers should run all of their application scripting logic to ensure the system creates any necessary cross-scope privilege records. After application publication, the system only allows runtime requests to run that have a valid cross-scope privilege record.

**Note:** Table privilege granting is limited to, at most, the permissions set on the table object (sys_db_object) record. For example, granting a scope privilege to delete for table incident would not be allowed if the table object for incident did not allow Can delete scopes.

**Application design access record**

Administrators use application design access records to specify which other applications are available to developers during application creation.

When administrators restrict an application's table choices, they can then create application design access records to grant developers access to the tables of selected applications.

Each application design access record contains the following information.

**Application design access fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Scope</td>
<td>The application requesting design access to another application's tables.</td>
</tr>
<tr>
<td>Target Package</td>
<td>The application whose tables will be available for design time access.</td>
</tr>
<tr>
<td>Application</td>
<td>The application to which this record belongs.</td>
</tr>
</tbody>
</table>

Application design access records allow administrators to have complete control of the resources available to application developers. When creating configuration records, application developers can only select tables from another application if there is an application design access record granting them access to the application. For example, administrators could create the following application design access records to grant developers access to tables from Incident Management and Problem Management.

**Sample application design access records**

<table>
<thead>
<tr>
<th>Source Scope</th>
<th>Target Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>My App</td>
<td>Incident</td>
</tr>
<tr>
<td>My App</td>
<td>Problem Management</td>
</tr>
</tbody>
</table>

After developers create configuration records to other applications, the system displays these applications as dependencies.

**Table design and runtime settings**

The Application access fields determine whether a table is accessible to other applications during design-time or run-time operations.
## Application access fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can read</td>
<td>Select the check box to allow script objects from other application scopes to read records stored in this table. This option offers runtime protection. For example, a script in another application can query data on this table. You must first select read access to grant any other API record operation.</td>
</tr>
<tr>
<td>Can write</td>
<td>Select the check box to allow script objects from other application scopes to modify records stored in this table. This option offers runtime protection. For example, a script in another application can modify a field value on this table. This option is available only when the Can read check box is selected. Clear the check box to prevent script objects from other application scopes from modifying data stored in this table.</td>
</tr>
<tr>
<td>Can create</td>
<td>Select the check box to allow script objects from other application scopes to create records in this table. This option offers runtime protection. For example, a script in another application can insert a new record in this table. This option is available only when the Can read check box is selected. Clear the check box to prevent script objects from other application scopes from creating records in this table.</td>
</tr>
<tr>
<td>Can delete</td>
<td>Select the check box to allow script objects from other application scopes to delete records from this table. This option offers runtime protection. For example, a script in another application can remove a record from this table. This option is available only when the Can read check box is selected. Clear the check box to prevent script objects from other application scopes from deleting records from this table.</td>
</tr>
<tr>
<td>Allow access to this table via web services</td>
<td>Select the check box to allow users to make inbound web service queries to this table. This option offers both design-time and runtime protection. The user performing the query must have the correct permissions to access this table, even when this check box is selected. Clear the check box to prevent users from making web service queries to this table.</td>
</tr>
</tbody>
</table>

### Runtime access to applications tables

Runtime access determines if an API or web service call can run against an application table. Access permissions can be set for the following access points.
The system does not prevent you from creating API or web service calls to the application tables, rather it determines if the API or web service call is allowed to run against the application table. API or web service calls that violate the access permissions for an application table produce an error. For example, making a web service call to a protected application table produces a 403 Forbidden HTTP error.

Default runtime access permissions
The default runtime access permissions apply to new application data tables.

By default, new application tables only allow read access from other application scopes.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessible from</td>
<td>All application scopes</td>
</tr>
<tr>
<td>Can read</td>
<td>Enabled</td>
</tr>
<tr>
<td>Can create</td>
<td>Disabled</td>
</tr>
<tr>
<td>Can update</td>
<td>Disabled</td>
</tr>
<tr>
<td>Can delete</td>
<td>Disabled</td>
</tr>
<tr>
<td>Allow access to this table via web services</td>
<td>Enabled</td>
</tr>
</tbody>
</table>
Set runtime access to application tables
Set these access permissions to protect application tables at runtime.

1. Navigate to System Applications > Applications.
2. Click the button for the application type you want to edit.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developed</td>
<td>displays applications created on this instance.</td>
</tr>
<tr>
<td>Downloaded</td>
<td>displays applications downloaded on this instance.</td>
</tr>
</tbody>
</table>

3. Click the application name or the Edit button for the application you want to work on.
4. From the Tables related list, select the table whose access permission you want to set.
5. From the Application Access section, set the runtime access permissions.
6. Click Update.

Example denying all runtime access to a table
You can prevent script API and web service calls from other application scopes.
Typically, this is to prevent any other application from creating or modifying data in the table.
Denying access requires setting the following value in the table record.
Denying all runtime access

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessible from</td>
<td>This application scope only</td>
</tr>
<tr>
<td>Can read</td>
<td>Disabled</td>
</tr>
<tr>
<td>Can create</td>
<td>Disabled</td>
</tr>
<tr>
<td>Can update</td>
<td>Disabled</td>
</tr>
<tr>
<td>Can delete</td>
<td>Disabled</td>
</tr>
<tr>
<td>Allow access to this table via web services</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

Limiting runtime access to this application scope only

The following diagram illustrates the effect of denying other application scopes access to application tables from script API and web service calls.
Deny all runtime access permissions to application tables

Example granting all runtime access to a table
You can permit some or all runtime script API and web service calls from other application scopes.
Granting access requires setting the following values in the table record.

```javascript
var gr = new GlideRecord("x_acme_book_rooms_conf_rooms");
gr.query("...
gr.update();
gr.insert();
gr.deleteRecord();
...}

Result: Policy refuses script access to table records.

REST Client Call
URL: instance/appserv/table/x_acme_book_rooms_conf_rooms
Headers: Accept: application/json, Content-Type: application/json,
X-UserToken: abcd1234...

Result: 403 Forbidden.
```
Granting all runtime access

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessible from</td>
<td>All application scopes</td>
</tr>
<tr>
<td>Can read</td>
<td>Enabled</td>
</tr>
<tr>
<td>Can create</td>
<td>Enabled</td>
</tr>
<tr>
<td>Can update</td>
<td>Enabled</td>
</tr>
<tr>
<td>Can delete</td>
<td>Enabled</td>
</tr>
<tr>
<td>Allow access to this table via web services</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

Granting other application scopes all runtime access permissions

The following diagram illustrates the effect of granting access to application tables from API calls and web services in other application scopes.
**ServiceNow**  Kingston  Now Platform Custom Business Applications

---

**App: Book Rooms**  
**Scope:** x_acme_book_rooms

- Allow API Operations?
  - Yes

**Conference Rooms**  
[x_acme_book_rooms_conf_rooms] table

- Allow Web Services?
  - Yes

**Web Services**
- REST
- JSON
- SOAP

---

**App: Marketing Events**  
**Scope:** x ITEC_marketing_events

- Script Call
  ```javascript
  var gr = new GlideRecord("x_acme_book_rooms_conf_rooms");
  gr.query();
  while (gr.next()){
    gr.update();
    gr.insert();
    gr.deleteRecord();
  }
  ```

- Result: Policy grants script access to table records.

- REST Client Call
  - URL: Instance/appinserttable/x_acme_book_rooms_conf_rooms
  - Headers: Accept:application/json, Content-Type:application/json, X-UserToken:abod1234...

- Result: 200 OK.

---

**Granted access to application tables**

*Design-time access to application tables*

As the application developer, you can grant or deny other applications the permission to create configuration records, also known as application files, that extend the functionality of an application.

The permission applies to any platform feature that extends the functionality of an application data table such as:

- Business rules
These access permissions protect the application data table at design-time. The system prevents you from creating configuration records by hiding the application data table as an option in the Table field. For example, a protected application table does not appear as an option when you create configuration records such as UI actions and client scripts.

Even when permission is granted to create configuration records, some configuration records have additional restrictions to protect application data from unwanted changes from other application scopes.

Default design access permissions
By default, new application tables prevent other application scopes from creating configuration records on application data tables. This prevents any other applications from changing the functionality of a table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessible from</td>
<td>All application scopes</td>
</tr>
<tr>
<td>Can read</td>
<td>Enabled</td>
</tr>
<tr>
<td>Can create</td>
<td>Disabled</td>
</tr>
<tr>
<td>Can update</td>
<td>Disabled</td>
</tr>
<tr>
<td>Can delete</td>
<td>Disabled</td>
</tr>
<tr>
<td>Allow access to this table via web services</td>
<td>Enabled</td>
</tr>
<tr>
<td>Allow configuration</td>
<td>Disabled</td>
</tr>
</tbody>
</table>
Default access permissions to configuration records

Set design-time access to application tables
Set these access permissions to protect application tables at design-time.

To set runtime access permissions:

1. Navigate to System Applications > Applications.
2. Click the button for the application type you want to edit.
   
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developed</td>
<td>displays applications created on this instance.</td>
</tr>
<tr>
<td>Downloaded</td>
<td>displays applications downloaded on this instance.</td>
</tr>
</tbody>
</table>
3. Click the application name or the Edit button for the application you want to work on.
4. From the Tables related list, select the table whose access permission you want to set.
5. From the Application Access section, set the design-time access permissions.
6. Click Update.

Example denying all design access to a table
You can prevent other application scopes from creating configuration records on application data tables.

Typically, this is to prevent any other applications from changing the functionality of a table. Denying access requires setting the following value in the table record.
Denying all design-time access

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessible from</td>
<td>This application scope only</td>
</tr>
<tr>
<td>Can read</td>
<td>Disabled</td>
</tr>
<tr>
<td>Can create</td>
<td>Disabled</td>
</tr>
<tr>
<td>Can update</td>
<td>Disabled</td>
</tr>
<tr>
<td>Can delete</td>
<td>Disabled</td>
</tr>
<tr>
<td>Allow access to this table via web services</td>
<td>Disabled</td>
</tr>
<tr>
<td>Allow configuration</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

Limiting design-time access to this application scope only

The following diagram illustrates the effect of denying other application scopes the ability to create configuration records.
Limiting design access to this application scope only

Example allowing configuration records for a table
You can permit other application scopes to create configuration records on application data tables.
You can grant access to the following configuration records with these settings.

Granting access to configuration records

<table>
<thead>
<tr>
<th>Configuration record</th>
<th>Setting required to grant access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access controls</td>
<td>• Accessible from set to All application scopes</td>
</tr>
<tr>
<td></td>
<td>• Can read is selected</td>
</tr>
<tr>
<td>Business rules</td>
<td></td>
</tr>
<tr>
<td>Client scripts</td>
<td>• Accessible from set to All application scopes</td>
</tr>
<tr>
<td>Dictionary entry (new field only)</td>
<td>• Can read is selected</td>
</tr>
<tr>
<td></td>
<td>• Allow configuration is selected</td>
</tr>
<tr>
<td>UI actions</td>
<td></td>
</tr>
</tbody>
</table>
Granting other application scopes design access permission

The following diagram illustrates the effect of granting other application scopes the ability to create configuration records.
Granting access to configuration records

Application restricted caller access settings

Restricted caller access settings enable an admin user to define cross-scope access to an application or application resource and allow or deny requests for access.

Records in the Restricted Caller Access (sys_restricted_caller_access) table track cross-scope applications or scripts that request access to an application or application resource. The system creates records when:

- Caller access is set to **Caller Restriction** or **Caller Tracking**.
- A cross-scope script attempts to access an application resource.

Using these records, an admin user or an application admin can:

- Track cross-scope requests for access to an application resource.
- Approve or deny any cross-scope requests for access to an application resource.

Alternatively, an admin can create a record in the Restricted Caller Access table to allow all scope-to-scope requests, or to allow all future requests to all application resources in the target scope.
If a calling resource changes, the restricted caller access record moves to an **Invalidated** status. An admin user or an application admin can update the status of the request accordingly.

**Activating application restricted caller access**

You can activate application restricted caller access through one of the following methods:

- Activate the Scoped Application Restricted Caller Access plugin (com.glide.scope.access.restricted_caller).
- Request the HR Service Delivery or Security Incident Response applications. These applications have application restricted caller access active by default.

**Activate application restricted caller access**

You can activate the Scoped Application Restricted Caller Access plugin (com.glide.scope.access.restricted_caller) if you have the admin role.

**Role required**: admin

1. Navigate to **System Definition > Plugins**.
2. Find and click the plugin name.
3. On the System Plugin form, review the plugin details and then click the **Activate/Upgrade** related link.
   - If the plugin depends on other plugins, these plugins are listed along with their activation status.
   - If the plugin has optional features that depend on other plugins, those plugins are listed under **Some files will not be loaded because these plugins are inactive**. The optional features are not installed until the listed plugins are installed (before or after the installation of the current plugin).
4. Optional: If available, select the **Load demo data** check box.
   - Some plugins include demo data—Sample records that are designed to illustrate plugin features for common use cases. Loading demo data is a good practice when you first activate the plugin on a development or test instance.
   - You can also load demo data after the plugin is activated by clicking the **Load Demo Data Only** related link on the System Plugin form.
5. Click **Activate**.

**Define cross-scope access to an application resource**

Admin users can track cross-scope requests for access to an application resource, and approve or deny requests.

**Role required**: admin or application admin

If application administration is enabled, only application administrators of the target application can set access to an application resource. If application administration is not enabled, an admin user can set access to an application resource.

1. To define access to an application resource, navigate to the application resource record.
   - Available application resources include:
     - Table
     - Script Include
2. Set the **Accessible from** field to **All application scopes**.
   - If set to **This application scope only**, no other application scopes can access the resource.
3. Select the appropriate access level in the **Caller Access** field.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Cross-scope calls to the resource are approved or denied based on the value of the Accessible from field.</td>
</tr>
<tr>
<td>Caller Restriction</td>
<td>Calls to the resource must be manually approved. Access requests are tracked in the Restricted Caller Access table with a status of Requested.</td>
</tr>
<tr>
<td>Caller Tracking</td>
<td>Calls to the resource are automatically approved. Calls are tracked in the Restricted Caller Access table with a status of Allowed.</td>
</tr>
</tbody>
</table>

4. Allow or deny an access request from a calling application.
   If a cross-scope application attempts to access a resource set to Caller Restriction, a record is created in the Restricted Caller Access table with a Requested status. An admin user or application administrator must allow or deny the request.

   If a calling resource changes, the restricted caller access record status changes to Invalidated. An admin user or application administrator must update the status to Allowed or Denied.

   a) In the application record, navigate to the Restricted Caller Access Privileges tab.
   b) In the Status column, set the value from Requested to Allowed or Denied.

   Once a calling script is allowed, all subsequent calls are allowed.

Define access to or from an application scope
To set access for the entire application, or to pre-approve or deny future requests from a cross-scope script or application, you can create a record in the Restricted Caller Access Privileges (sys_restricted_caller_access) table.

Role required: admin or application admin

If application administration is enabled, only application administrators of the target application can set access to an application. If application administration is not enabled, an admin user can set access to an application.

1. Navigate to System Applications > Application Restricted Caller Access.
   The Restricted Caller Access Privileges (sys_restricted_caller_access) table opens.
2. Complete the form.

   To allow access between two application scopes, set the Source Type and Target Type to Scope and select the appropriate Source Scope and Target Scope.
### Restricted Caller Access fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>Operation on the target resource.</td>
</tr>
<tr>
<td></td>
<td>• Read</td>
</tr>
<tr>
<td></td>
<td>• Write</td>
</tr>
<tr>
<td></td>
<td>• Create</td>
</tr>
<tr>
<td></td>
<td>• Delete</td>
</tr>
<tr>
<td></td>
<td>• Execute API</td>
</tr>
<tr>
<td>Source</td>
<td>Record of the calling script.</td>
</tr>
<tr>
<td>Source Scope</td>
<td>Scope of the calling application.</td>
</tr>
<tr>
<td>Source Table</td>
<td>Table that contains the <strong>Source</strong> record.</td>
</tr>
<tr>
<td>Source Type</td>
<td>Type of record calling the application resource.</td>
</tr>
<tr>
<td></td>
<td>• Scope</td>
</tr>
<tr>
<td></td>
<td>• Business Rule</td>
</tr>
<tr>
<td></td>
<td>• UI Action</td>
</tr>
<tr>
<td></td>
<td>• Script Include</td>
</tr>
<tr>
<td></td>
<td>• Scheduled Script</td>
</tr>
<tr>
<td></td>
<td>• Workflow Activity</td>
</tr>
<tr>
<td></td>
<td>• Service Portal Widget</td>
</tr>
<tr>
<td></td>
<td>• UI Macro</td>
</tr>
<tr>
<td></td>
<td>• Orchestration RunScript Activity</td>
</tr>
<tr>
<td></td>
<td>• Document Title</td>
</tr>
<tr>
<td></td>
<td>• GlideScopedEvaluator</td>
</tr>
<tr>
<td></td>
<td>• ACL</td>
</tr>
</tbody>
</table>

To allow access from an entire application, select **Scope**.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Status of the access request.</td>
</tr>
<tr>
<td></td>
<td>• Requested</td>
</tr>
<tr>
<td></td>
<td>• Denied</td>
</tr>
<tr>
<td></td>
<td>• Allowed</td>
</tr>
<tr>
<td></td>
<td>• Invalidated</td>
</tr>
<tr>
<td>Note:</td>
<td>If a calling resource changes, the restricted caller access record status changes to Invalidated.</td>
</tr>
<tr>
<td></td>
<td>If application administration is enabled, only application administrators of the target application can update the status of a request.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Target</td>
<td>Record of the requested resource.</td>
</tr>
<tr>
<td>Target Scope</td>
<td>The scope of the requested resource.</td>
</tr>
<tr>
<td>Target Table</td>
<td>Table that contains the Target record.</td>
</tr>
<tr>
<td>Target Type</td>
<td>Type of requested resource.</td>
</tr>
<tr>
<td></td>
<td>• Scope</td>
</tr>
<tr>
<td></td>
<td>• Table</td>
</tr>
<tr>
<td></td>
<td>• Script Include</td>
</tr>
<tr>
<td></td>
<td>To allow access to an entire application, select Scope.</td>
</tr>
</tbody>
</table>

**Application list**

The applications list allows application developers to view and select applications.

Application developers can use the applications list to open a custom application record. If the contextual development environment detects that you are editing an application artifact in another application scope, it displays a warning message you can use to switch to another application.

Administrators have the following options from the applications list.
Sample list of applications

Application list options

<table>
<thead>
<tr>
<th>Tab name</th>
<th>Options available</th>
<th>Application source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop</td>
<td>• Create new application&lt;br&gt;• Edit existing application&lt;br&gt;• Share application&lt;br&gt;• Delete application</td>
<td>Applications developed on this instance</td>
</tr>
<tr>
<td>Downloads</td>
<td>• Install&lt;br&gt;• View installed files&lt;br&gt;• Edit installed files&lt;br&gt;• Uninstall</td>
<td>• Applications shared on the company application repository&lt;br&gt;• Applications shared on the ServiceNow Store</td>
</tr>
<tr>
<td>Updates</td>
<td>• Update&lt;br&gt;• View version information</td>
<td>• Applications shared on the company application repository&lt;br&gt;• Applications shared on the ServiceNow Store</td>
</tr>
</tbody>
</table>

The application list displays the following information.

Applications list

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developed</td>
<td>Display applications created on this instance.</td>
</tr>
<tr>
<td>Downloaded</td>
<td>Display applications downloaded on this instance.</td>
</tr>
<tr>
<td>Updates</td>
<td>Display available updates for downloaded applications on this instance.</td>
</tr>
</tbody>
</table>
Application picker

The application picker allows application developers to view and select the application where their changes apply.

The application picker is available from the system menu under the gear icon in the banner frame.
Lists and forms in scoped applications

The current application context determines what customization and form design options are available when working with lists and forms in scoped applications.

The user interface uses visual indicators to identify a list or form in the same or different application scope.

Available layout and design actions

The system allows the following layout and design actions when working on lists or forms in custom applications.

| Available layout and design actions |
|------------------------------------|-------------------------|
| **Action**                         | **Access granted**      | **Notes**                                                   |
| Create a list view                 | Always allowed          | This action is always available to users with access to customization. |
| Create a form view                 | Always allowed          | This action is always available to users with access to customization. |
| Create a form section              | Always allowed          | This action is always available to users with access to customization. |
| Select fields to display in a view | Only allowed for sections in the view that match the current scope. | This restriction is independent of a user’s role. Administrators cannot bypass this restriction. |
| Change the order of sections in a view | Only allowed for views that match the current scope. | This restriction is independent of a user’s role. Administrators cannot bypass this restriction. |
| Select fields to display in a section | Only allowed for sections that match the current scope. | This restriction is independent of a user’s role. Administrators cannot bypass this restriction. |
| Add or remove section columns      | Only allowed for sections that match the current scope. | This restriction is independent of a user’s role. Administrators cannot bypass this restriction. |
| Delete a form section              | Only allowed for sections that match the current scope. | This restriction is independent of a user’s role. Administrators cannot bypass this restriction. |
| Create new fields                  | Only allowed for sections that match the current scope and when the **Allow configuration** option is enabled. | This restriction is independent of a user’s role. Administrators cannot bypass this restriction. |

Form design visual indicators

The UI displays the following visual indicators when designing forms in custom applications.

You can only edit views and sections when you are in the same application scope as the form. Editable sections display:

- Section headings with a solid color background.
- A solid line around the section.
- A control to set the number of columns.
- A **Delete this section** button.
- Grip icons beside section headings.
- Grip icons beside fields.

**Visual indicators of editable sections**

Views and sections in another application scope display as read only. Read-only sections have:

- Section headings with a gray background.
- A gray line around the section.
- No control to set the number of columns.
- No **Delete this section** button.
- No grip icons beside section headings.
- No grip icons beside fields.
Visual indicators of read-only sections

Default form design permissions
By default, new application data tables have the following form design permissions.

<table>
<thead>
<tr>
<th>Form design action</th>
<th>Permission setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create new sections in tables belonging to another application scope</td>
<td>Allowed</td>
</tr>
<tr>
<td>Create new fields in sections belonging to the same application scope</td>
<td>By default, denied. Requires application designer to set Allow configuration for application table.</td>
</tr>
<tr>
<td>Add or remove fields from sections belonging to the same application scope</td>
<td>Allowed</td>
</tr>
<tr>
<td>Change the order of fields in sections belonging to the same application scope</td>
<td>Allowed</td>
</tr>
<tr>
<td>Change the order of sections belonging to the same application scope</td>
<td>Allowed</td>
</tr>
<tr>
<td>Form design action</td>
<td>Permission setting</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Add or remove fields from sections belonging to another application scope</td>
<td>Denied</td>
</tr>
<tr>
<td>Change the order of fields in sections belonging to another application scope</td>
<td>Denied</td>
</tr>
<tr>
<td>Change the order of sections belonging to another application scope</td>
<td>Denied</td>
</tr>
<tr>
<td>Create new fields in sections belonging to another application scope</td>
<td>Denied</td>
</tr>
</tbody>
</table>

**Layout visual indicators**

The UI displays the following visual indicators when configuring the layout of a custom application's list or form.

You can only edit fields, views, and sections when you are in the same application scope as the form. Editable sections display:

- Field selections with a white background.
- Buttons to add or remove fields.
- Save button with a blue background.
- Editable fields in the **Create new field** section.
### Configuring Table Form

**Available**
- Affected by Release
- Affected by Task
- Agreements
- Application PID -> CMDB CI
- Approval group [+]
- Asset [+]
- Assigned
- Assigned to [+]
- Assignment Data Lookup -> Configuration Item
- Assignment group [+]
- Attributes
- BSM Saved Map -> Configuration Item
- Can Print
- Catalog Task -> Configuration Item
- Category

**Selected**
- Name
- Location
- Capacity

### Form view and section

**View name**: Default view

**Section**
- Conference Rooms
  - Room Resources
    - New...

### Create new field

- **Name**: __________
- **Type**: String
- **Field length**: Small (40)

### Related Links

- Show versions
Views and sections in another application scope display as read only. Read-only sections have:

- A warning message about the currently selected application scope and the scope of the form.
- Field selections with a gray background.
- No buttons to add or remove fields.
- Save button with a gray background.
- Read only fields in the **Create new field** section.
The 'Conference Rooms' section is in the Book Rooms application, but Marketing Events is selected in your application picker. To edit this form:
- Create a new Marketing Events section
- Select an existing Marketing Events section
- Switch to Book Rooms to edit this section

**Available**
- Approval group [+]
- Asset [+]
- Asset tag
- Assigned
- Assigned to [-]
- Assignment group [+]
- Attributes
- Can Print
- Category
- Checked In
- Checked Out
- Class
- Comments
- Company [+]
- Correlation ID
- Cost
- Cost center [+]
- Cost currency
- Created

**Selected**
- Name
- Location
- Capacity

### Form view and section
- **View name**: Default view
- **Section**: Conference Rooms

### Create new field
- **Name**: [Field Name]
- **Type**: String
- **Field length**: Small (40)
Contextual development edit messages

The platform displays a message whenever a user attempts to edit an application artifact belonging to a different application.

This record is in the Marketing Events Application application, but Global is the current application. To edit this record click here.

Application context edit message

This message can be used to:

- Open the application to which the current configuration record belongs.
- Open the application of the currently selected application in the application picker.
- Temporarily switch to the application to which the current configuration record belongs and edit it.

Note: The system returns you to the current application context after you save or cancel out of the record.

The system also displays a message when a user attempts to configure a list or form layout while working from another application scope.

The 'Incident' section is in the Global application, but Marketing Events Application is the current application. To edit this form:

- Edit this section in Global
- Create a section in Marketing Events Application
- Create a view in Marketing Events Application

Note: The system returns you to the current application context after you save or cancel out of the record.

Application context edit message for form layout or design

The message provides a list of valid options:

- Edit the current section by temporarily switching to the application that owns it.
- Create a new section in the current application context.
- Create a new view in the current application context.

Note: The system returns you to the current application context after you save or cancel out of the record.

Application resource throttling

The system uses a set of transaction quotas to limit the resources that any one custom application can consume.

These limits are intended to prevent custom applications from exhausting instance resources and causing an outage.
Default transaction quota rule

By default, the following transaction quota limits apply to custom applications.

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Transaction quota limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business rules (foreground)</td>
<td>5,000</td>
</tr>
<tr>
<td>Events (foreground)</td>
<td>5,000</td>
</tr>
<tr>
<td>Jobs (foreground)</td>
<td>5,000</td>
</tr>
<tr>
<td>Outbound HTTP requests (foreground)</td>
<td>10</td>
</tr>
<tr>
<td>Outbound HTTP request duration (foreground)</td>
<td>100 seconds</td>
</tr>
</tbody>
</table>

Script protection policy

Application developers can set a protection policy for script includes published as part of a custom application. The policy determines whether someone can view or edit the script include after the application is installed on their instance.

Application developers have these options to protect their custom application script includes:

<table>
<thead>
<tr>
<th>Protection policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Allows anyone who installs your published application to view and modify this script include on their instance. Select this option to allow other application developers to customize your script include.</td>
</tr>
<tr>
<td>Read-only</td>
<td>Allows anyone who installs your published application to view this script include on their instance. Select this option to allow other application developers to see your script logic, but not to change it.</td>
</tr>
<tr>
<td>Protected</td>
<td>Prevents anyone who installs your published application to view or modify this script include on their instance. Select this option to prevent other application developers from seeing or changing your intellectual property.</td>
</tr>
</tbody>
</table>

Application administration

Protect sensitive application data by using application administration to restrict how users acquire application-specific roles.

Functions of application administration

Use application administration to:

- Prevent unauthorized users from accessing sensitive data such as financial records or personally identifiable information.
- Restrict who can assign application-specific roles such as the administrator and designated developers for the application.
- Prevent users with the system-level admin role from:
  - Assigning themselves a protected application role.
  - Assigning themselves to a group containing a protected application role.
• Bypassing existing access controls to a protected application by creating access controls.
• Changing the password of users who have a protected application role.
• Impersonating a user who has a protected application role, unless the developer or administrator also has that role.
• Inheriting a protected application role.
• Overriding existing access controls to a protected application.
• Running scripts that access protected application records.

Roles in application administration

You can make any role an application-specific administrator by selecting the **Application Administrator** check box in Role Configuration. To learn more, see *Enable application administration*. By convention, create the following roles:

**Application administration roles**

<table>
<thead>
<tr>
<th>Role name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application-specific admin</td>
<td>Users with this role can assign other users to an application-specific role for that application.</td>
</tr>
<tr>
<td></td>
<td>For example, you can create a role named my_application.admin. It should include the name of the restricted application, with a suffix of</td>
</tr>
<tr>
<td></td>
<td>“admin” to indicate that it is the admin role for the application.</td>
</tr>
<tr>
<td>Application-specific developer</td>
<td>Users with this role can access the restricted application. For example, you can create a role named my_application.developer. It should</td>
</tr>
<tr>
<td></td>
<td>include the name of the restricted application, with a suffix of “developer” to indicate that it is the developer role for the</td>
</tr>
<tr>
<td></td>
<td>application. The developer role needs both application administrator and delegated development permissions to modify the application files.</td>
</tr>
<tr>
<td></td>
<td>To learn more, see <em>Delegated development</em> and <em>Add a developer</em>.</td>
</tr>
</tbody>
</table>

**Application-specific admin role**

The application-specific admin role enables a user to access a specific application, but does not grant the user any other admin rights. Assign the system-level admin role to a user before that user can do these tasks:

• Configure form and list layouts.
• Change application tables and fields.
• Assign the application-specific admin role to new users.

If you do not want a user with the application-specific admin role to have the system-level admin role:
- Do not assign the system-level admin role to the user. Assign only the application-specific admin role.
- Have the user assign themselves the application-specific developer role.

As an application-specific developer, the user can perform a subset of administrative tasks without having the system-level admin role.

**Note:** Assign the application-specific admin role to more than one user. Then if a user with the application-specific admin role leaves the company, you are not prevented from changing the application.

### Enabling application administration and assigning application-specific roles

You can enable application administration for an application from the application record and restrict the assignment of application-specific roles from the user role record.

**Note:** Enable application administration and assign application-specific roles after completing development of the application, but before adding application records. This practice protects sensitive data in the application records from access by unauthorized users.

The target instance must have at least one authorized user with the application-specific admin role.
- If you enable application administration for an application but do not assign the application-specific roles, no user can access the application.
- If you assign only one application-specific role, you cannot delete that role.

### Training

The ServiceNow Developer Site has training for *Securing Applications*.

### Enable application administration

Enabling application administration restricts both an application’s content and its user management to designated application administrators.
- Role required: admin
- Records required:
  - Application
  - User

1. Optional: If necessary, create an application administration role.
   For example, create a role with the suffix admin. The actual role combines the application scope with the suffix such as x_my_app.admin.
2. Open the role record for the application admin role.
3. Configure the form to add the **Application Administrator** field.
4. Select **Application Administrator**.
Note: The **Assignable by** field is no longer in use and is replaced by the **Application Administrator** check box. On upgrade, any role in the **Assignable by** field will have **Application Administrator** selected by default.

5. Click **Update**.

6. Log out and log back in as the application administrator.
   Only the application administrator can enable application administration for an application.

7. Navigate to **System Applications > Applications**.

8. Click the name of the application for which you want to enable application administration.
   The system opens the application record.

9. Select **Application administration**.

10. Click **Update**.
    The system validates the following requirements have been met.
    - The application has an admin role (there is at least one role with **Application Administrator** selected).
    - The current user has the application admin role.

    If the validation passes, the system updates the application record. Otherwise, the system displays an error message.

11. Optional: From **Related Links**, you can select one of the following options:

<table>
<thead>
<tr>
<th>Related Link</th>
<th>Description</th>
</tr>
</thead>
</table>
| Manage Developers | The Manage Developers modal opens. The application administrator can:  
- Designate developers for the application.  
- Make themselves a delegated developer. Once a delegated developer, the application administrator can perform a subset of administration tasks without having the admin role.  
Learn more: *Delegated development* |
| Grant application administration to all admins | The system creates a Contained Role (sys_user_role_contains) record for the admin role, which adds the application admin role as a contained role of the admin role.  
Note: Publishing the application with this record allows users with the admin role to have access to application after installing it. |

**Deploy an application administration app**

Deploying an application administration application requires the target instance have at least one authorized user with the proper application admin role.

- **Role required:** The application’s admin role
- **Records required:**
  - Application
  - User with application’s admin role
This process assumes that your developer and production instances share at least one admin user.

1. Develop the application on a development instance.
2. Create the application admin role.
3. Grant application administration to all admins.
4. Update the application to enable application administration.
5. Publish the application to the application repository.
6. From a production instance, install the application from the application repository.
7. As an admin on production, grant the application’s admin role to the appropriate users.
8. Remove application administration from admins.

Access control rules in application administration apps

By default, when application administration is enabled for a scoped application, ACL rules for the scoped application are applied. If no ACL rules for the scoped application are found, global ACL rules can apply.

This behavior applies to configuration records created in tables that extend the Application File (sys_metadata) table only. You can also change the default behavior.

When no access control (ACL) rules for an application administration app are defined, global ACL rules can apply to the configuration records of the application administration apps. See Application files for more information.

To allow a table in an application administration app to inherit global ACL rules, check that the system property is true and add the table to the Application Administration ACL Inheritances table (sys_scoped_admin_acl_inheritance).

- glide.security.scoped_administration.honor_global_acl system property: If no scoped ACL rules are defined, application administration apps can inherit global ACL rules. By default, this property is enabled for new and upgraded instances.
- Application Administration ACL Inheritances (sys_scoped_admin_acl_inheritance) table: If no ACL rules for the application administration app are found, tables added to this list inherit global ACL rules. Only the administrator for the application administration app can add, remove, read the records owned by the application administration app in this configuration table.

Configure a table in an application administration app to inherit global ACL rules

To avoid duplicating global access control rules (ACLs) in your applications, you can configure application file tables in application administration apps to inherit global ACLs when no ACL rules for the scoped application are found.

Role required: admin role for the application

1. Enter sys_scoped_admin_acl_inheritance.list in the navigation filter. The Application Administration ACL Inheritance table (sys_scoped_admin_acl_inheritance) opens.
2. Click New.
3. In the Table field, select an application file table (a table that extends the Application File (sys_metadata) table) from the application administration app. Only the administrator for the application administration app can add/remove/read records owned by the application administration app in this configuration table.
4. Click Submit.

If no ACL rules from the application administration app are found, tables added to this list inherit global ACL rules.
Access enforcement for ServiceNow Store apps

All production instances monitor and generate reports on usage patterns for ServiceNow Store apps. When subscription enforcement is enabled, users who are not subscribed to the app are blocked from performing fulfiller actions in the app.

Overview

The following actions are required to enable a production instance to enforce entitled usage of your ServiceNow Store App:

1. The usage admin at your organization uses the Subscription Management application to allocate fulfiller users to the subscription.
2. You decide on the enforcement mode, either:
   - Monitor and report usage with no enforcement (default)
   - In addition to monitoring and reporting usage, enforce that all usage must be by subscribed fulfiller users.
3. To enforce usage only by subscribed users, you configure the tables where only record owners or subscribed fulfiller users can make updates as fulfillment tables.

Creating applications

The Now Platform allows you to create applications to meet your business needs.

When building applications on the ServiceNow platform, application developers can take advantage of:

- Contextual development environment
  - ServiceNow Studio
  - Application scope
  - Script protection policy
  - Source control integration
- Platform features
  - Security access controls
  - Workflow
  - Business rules
  - APIs
  - Notifications
- Shared system data
  - User records
  - Task records
  - Application records made accessible to other applications

Video: creating an application

Watch a video demonstration of creating a visitor registration application.
You can create this example application by registering for your own demonstration or developer instance.

**Basic development process**

You can use this basic development process to create quality applications.

### Basic application development process

<table>
<thead>
<tr>
<th>Process</th>
<th>Description</th>
<th>Recommended tools</th>
</tr>
</thead>
</table>
| 1. Define business requirements. | Create business requirements and track their progress. Your requirements should answer questions such as:  
- What does the application need to do?  
- Who will use it?  
- How will you know if it works? | •  
•  |
| 2. Define the data model.        | Identify what information the application needs to track.  
- What information needs to be captured as a record?  
- What references are there between records?  
- Are there any existing tables you can extend? | ServiceNow Studio |
<table>
<thead>
<tr>
<th>Process</th>
<th>Description</th>
<th>Recommended tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Build the application.</td>
<td>Create application and configuration records.</td>
<td>• ServiceNow Studio</td>
</tr>
<tr>
<td></td>
<td>1. Select an application creation method to create a custom application record and set the application scope.</td>
<td>• Team Development</td>
</tr>
<tr>
<td></td>
<td>2. Create application data tables to store application-specific data.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Design the user interface, such as the list and form layout.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Set application access settings to permit or restrict other applications from accessing application data.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. Add business logic and automation such as business rules, events, and workflows, to meet your business requirements.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Support multiple developers working on the application.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Push the application to other instances in the team development environment.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Link the application to a GIT repository to save and manage multiple versions.</td>
<td></td>
</tr>
<tr>
<td>4. Test the application.</td>
<td>Verify the application meets your business requirements. Your testing should cover:</td>
<td>•</td>
</tr>
<tr>
<td></td>
<td>• Record operations such as create, read, update, and delete.</td>
<td>•</td>
</tr>
<tr>
<td></td>
<td>• User interface elements such as views and UI policies.</td>
<td>•</td>
</tr>
<tr>
<td></td>
<td>• Runtime operations such as business rules and event script actions.</td>
<td>•</td>
</tr>
<tr>
<td>Process</td>
<td>Description</td>
<td>Recommended tools</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Share the application.</td>
<td>Share the completed application with others.</td>
<td><strong>ServiceNow Studio</strong></td>
</tr>
<tr>
<td></td>
<td>* Publish the application to the ServiceNow application repository to share it with other instances assigned to the same company.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* Publish the application to the ServiceNow Store to share it with everyone.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* Publish the application as an Update Set to comply with a change management or backup policy.</td>
<td></td>
</tr>
</tbody>
</table>

**Application creation options**

Application developers have options when creating applications.
Sample application creation options

To start creating an application, navigate to System Applications > Studio and click Create Application. Select one of the available options.

- Start from scratch
- Create custom application
- Start from a template
- Start from an existing service (available only when the Service Creator plugin is activated)
- Start from global (only displays if the glide.app.creator.global is set to true).

The application creation method determines the application scope.

After you create an application, it appears on the applications list (System Applications > Applications) and is set as the current application in the application picker.

Start from scratch

You can create an empty application using the Start from scratch option.
This option is good for applications that only use one configuration record, such as a workflow or script-based application.

1. From the **Start from scratch** row, click **Create**.
2. Enter a **Name** for the application.
3. Create **Create**.
4. In the Confirm Application dialog box, click **OK**.

Create a custom application

You can create an application with UI and data elements using the **Create custom application** option.

This option is good for applications that require UI elements and tables.

1. From the **Create custom application** row, click **Create**.
2. Enter a **Name** for the application.
3. Optional: Modify any of these fields if the default values do not meet your needs.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu</td>
<td>The application menu users can use to access the application.</td>
</tr>
<tr>
<td>User Role</td>
<td>The role a user must have to access the application.</td>
</tr>
</tbody>
</table>

4. Optional: Select **Create Table** and provide table details.

If you do not create a table using this option, you must manually define a table before you can store any application data.

5. Click **Create**.
6. On the Confirm Application dialog box, click **OK**.

The system creates the application and provides an **Edit App** option.

Start from a template

You can create service management applications from a template using the **Start from a template** option if service management is active.

This option is good for applications that need to track the status of requests or task-driven records.

1. From the **Start from a Template** row, click **Create**.
2. Enter a **Name** for the application.
3. Optional: Click **Configure** from the section containing the process you want to use, and select the configuration options you want to use.
4. Click **Create** from the section containing the process you want to use.
5. In the Confirm Application dialog box, click **OK**.

The system creates the application and provides an **Edit App** option.

Start from an existing service

You can convert an existing service to an application if service creator is active.

You may want to convert successful services into full applications to take advantage of features only available to applications such as contextual security, data or UI policies, UI actions, and auditing.
When converting a service that exists on a production instance, you should pull the service to a non-production instance and convert the service to an application in that environment. After completing the conversion, you can push the new application to the production environment.

1. From the **Start from a service** row, click **Create**.
2. In the **Create from service** field, select a published service.
3. Optional: Change the application details and catalog item details if the default values do not meet your needs.
4. Optional: Select **Replace original service** to deactivate the catalog item for the service and replace it with an equivalent catalog item for the application.

When replacing an existing service with an application, the application preserves the same end-user experience for submitting and fulfilling requests. Any service requests that were made prior to the conversion are converted to requests within the application.

5. Click **Create**.
6. Click **OK**.

**Conversion mapping**
The conversion process uses the following service creator elements to create application records.

<table>
<thead>
<tr>
<th>Service Creator element</th>
<th>Becomes this application element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service category table</td>
<td>Application table</td>
</tr>
<tr>
<td>Service name</td>
<td>Table label</td>
</tr>
<tr>
<td>Service form layout</td>
<td>Application table form layout</td>
</tr>
<tr>
<td>Service catalog policies</td>
<td>Application table UI policies</td>
</tr>
<tr>
<td>Service catalog item</td>
<td>Application catalog item</td>
</tr>
<tr>
<td>Service catalog variable</td>
<td>Fields on the application table</td>
</tr>
<tr>
<td>Category form layout</td>
<td>Table form layout</td>
</tr>
<tr>
<td>Service task workflow</td>
<td>Application table workflow</td>
</tr>
</tbody>
</table>

**Start from global**

You can create an application in the global scope using the **Start from global** option.

Role required: admin

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Menu</strong></td>
<td>The application menu users can use to access the application.</td>
</tr>
</tbody>
</table>
Create an application data table

Create one or more custom tables to store application-specific data. The relationships between these tables form the application data model.

1. Navigate to System Applications > Applications.
2. Open the application record.
3. Scroll to the Tables related list.
4. Click New or use the list controls to create a new application data table.
5. Define the table and columns by completing the form.

Design the user interface

The system automatically creates a list and form view for each application data table. You can use the standard system processes to lay out these lists and forms to add views, fields, and sections as needed.

1. Navigate to System Applications > Applications.
2. Open the application record.
3. Scroll to the Tables related list.
4. Select the table for which you want to configure the UI.
5. Select one of the following related links to configure the UI:
   - Design Form
   - Layout Form
   - Layout List
   - Show Form
   - Show List
6. Optional: Create custom menus, modules, or UI pages to allow users to find, add, and modify the information they need.

Add a related list to another application’s table view

Add a related list from any application to a table view to help users find related records in newly installed applications.

You must have created a custom application that has a relationship to data in another application. For example, a new application table with a reference to incidents or configuration items.

Role required: admin
While you can always create a new view for another application's form, sometimes you want to increase the visibility of your application's data by adding a related list to an existing form. For example adding a related list to the end of a table's default view.

1. Switch to your application.
2. Navigate to the form view you want to add a related list to.
3. Right-click the form header and click **Configure > Related Lists.**
The system displays the Related Lists form.

4. Click the **Create a relationship** link in the information message. The system displays the Relationship form with the **Applies to table** field automatically set to the table.
5. If your application already contains a reference field to the table, click **Simple reference** otherwise go to **step 8**.

6. Enter the following field values.

**Relationship field values for a simple reference**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Any unique name</td>
</tr>
<tr>
<td>Queries from table</td>
<td>Select the application table containing the reference field.</td>
</tr>
<tr>
<td>Reference field</td>
<td>The system automatically populates this field with the reference field to the table you are adding the related list to.</td>
</tr>
</tbody>
</table>

7. Click **Submit**.
   The system automatically appends the relationship as the last related list entry on form view you started from.
8. For more complex relationships, enter the following field values.
### Relationship field values

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Any unique name</td>
</tr>
<tr>
<td>Applies to table</td>
<td>Select the table on which the related list appears.</td>
</tr>
<tr>
<td>Queries from table</td>
<td>Select the table from which this related list retrieves data.</td>
</tr>
<tr>
<td>Queries with</td>
<td>Type a script to specify the records to include from the table the relationship queries.</td>
</tr>
</tbody>
</table>

9. Right-click the form header and click **Save**.  
The system displays the **External Related Lists** related list.

10. From **External Related Lists**, click **Edit**.
11. Select an existing related list view or views you want to append your related list to.
12. Click **Save**.  
The system appends the relationship as the last related list entry on form view or views you selected.

### Set application access

Define what data other applications can access from this application.

By default, new applications have the following design and runtime access settings.
- Allow scripts runtime access other applications with Runtime Access Tracking.
- Allow developers to create configuration records for tables from other applications.
- Allow administrators to access application data without scoped administration.

By default, new application tables have the following design and runtime access settings:
- Allow runtime access to read table data.
- Allow web service access to table data.
- Hide the application table during design-time configuration.

1. Navigate to **System Applications > Applications**.
2. Open the application record.
3. From the **Design and Runtime** section, set the **JavaScript Mode**, **Runtime Access Tracking** and **Restrict Table Choices** options as necessary.
4. Scroll to the **Tables** related list.
5. Select the table for which you want to configure access settings.
6. From the Application Access section, set the runtime and design-time access permissions:
   - Run-time settings: determine whether the system allows or denies access to the application data tables at run-time.
   - Design-time settings: determine whether other application developers can create application files for your application table.
7. Click **Update**.
Add application logic

Add application-specific business logic by creating application files in existing platform tables. For example, add business rules, script includes, workflows, and UI actions. Only the application tables that allow configuration can have application files.

1. Determine which application tables need application files to fulfill application logic requirements.
2. Select the Allow configuration option on the table record for each of these tables.
3. Create the application file you want for the table. For example, create a business rule that runs on an application table.
4. Test your application logic.

Creation restrictions across application scopes

The system restricts the creation of some configuration records when the current application scope does not match the application scope of the configuration record’s target table.

Configuration record creation restrictions prevent one application from making unwanted changes to another application’s data tables. These restrictions only apply when you create a configuration record whose target table belongs to another application. Configuration records that belong to the same application scope do not have these restrictions.

The system always enforces the following creation restrictions when a developer adds a configuration record belonging to another application scope.

<table>
<thead>
<tr>
<th>Configuration record type</th>
<th>Creation restrictions when target table is in another application scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access controls</td>
<td>• You can only create field-level access controls with a role-based requirement.</td>
</tr>
<tr>
<td></td>
<td>• You cannot create table-level access controls for a table in another application scope.</td>
</tr>
<tr>
<td></td>
<td>• You cannot create field-level access controls that apply to all fields.</td>
</tr>
<tr>
<td></td>
<td>• You cannot create access controls that use conditions.</td>
</tr>
<tr>
<td></td>
<td>• You cannot create access controls that use a script-based condition.</td>
</tr>
<tr>
<td>Configuration record type</td>
<td>Creation restrictions when target table is in another application scope</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Business rules            | • You can create a rule where **When** is **async** with any of the following options:  
  • Insert, Update, and Delete database operations. You cannot select **Query**.  
  • Set field values actions and scripts (the **Script** field).  
  • You can create a rule where **When** is **before** with any of the following options:  
    • Insert, Update, and Delete database operations. You cannot select **Query**.  
    • Set field values actions only. You cannot write scripts and you cannot abort the database transaction.  
  • You cannot create any other type of business rules on tables in a different scope. |
| Calculated fields         | You cannot create calculated fields for tables in another application scope. |
| Data policies             | • You cannot create data policy rules for fields in another application scope.  
  • You cannot make a field mandatory. |
| Field styles              | You cannot create field styles for fields in another application scope. |
| Form sections             | • You cannot modify existing form sections created in another application scope.  
  • You can create new form sections. |
| Record producers          | You must have create access to the application table to create records from a record producer. |
| UI policies               | • You cannot create UI policy rules for fields in another application scope.  
  • You cannot make a field mandatory. |
| UI script                 | You cannot create a global UI script from a scoped application. |
| Views                     | • You can create new views.  
  • You cannot modify existing views created in another application scope. |

**Delegated development**

Delegated development allows users without the system admin role to develop applications on the ServiceNow platform.

Application administrators or system admins can manage designated developers by specifying which users and groups have access to application content.
- If *Application administration* is enabled, only an application administrator for the target application can delegate developers to an application. Application administrators do not have system admin privileges.
- If application administration is not enabled, a system admin can delegate developers.

**Sample developer permissions**

For each application, application administrators can:
Grant non-system admin users the ability to develop applications.
Specify which application file types the developer can access.
Grant the developer access to security records.
Grant the developer access to script fields.
Remove a user or group as a developer.

**Important:** To enable a delegated developer to perform the functions granted in the developer permissions, the delegated developer must also be given the application administrator role.

Each developer permission grants one or more delegated-development-specific roles. These roles allow system admins to retain control over the system, since they no longer have to elevate developers to the system admin role to enable the developers to create applications.

Although system admins can still manually assign and remove roles, they are encouraged to let the system manage the following delegated developer roles.

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delegated_developer</td>
<td>Indicates that the user has one or more developer permissions.</td>
</tr>
<tr>
<td>Roles that start with an sn_dd prefix</td>
<td>Indicates that this user has an application-specific developer permission. The role name indicates the application scope to which it applies.</td>
</tr>
</tbody>
</table>

**Note:** Delegated development cannot add or remove the admin role.

Developer permissions are application-specific. For example, a developer who has permission to access all file types for one application does not necessarily have any developer permissions for another application. Application administrators must set developer permissions for each application.

Application administrators must be familiar with application files and the system table structure to set developer permissions. For example, a developer expected to create advanced business rules needs both the All File Types and Allow Scripting developer permissions.

**Add a developer**

Adding a user or group as a developer allows the developer to create and change applications.

- Role required: admin or application administrator
  - If Application administration is enabled, only an application administrator of the target application can delegate developers to an application. If application administration is not enabled, an admin user can delegate developers.
- Records required:
  - Application
  - User
  - Group

1. Navigate to **System Applications > Applications**.
2. Click the application name of the application to which you want to add developers. The system opens the application record.

3. Click **Manage Developers**. The system displays the Developer Permissions window.

4. Select the type of developer you want to create.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developers</td>
<td>To add individual users as developers.</td>
</tr>
<tr>
<td>Groups</td>
<td>To add all members of a group as developers.</td>
</tr>
</tbody>
</table>

5. In **developer name** or **developer group name**, enter the name of the user or group you want to grant developer permissions.

6. Select permissions for the developer.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All File Types</td>
<td>Grants the developer access to all application file types including some not granted by the other options.</td>
</tr>
<tr>
<td>Integrations</td>
<td>Grants the developer access to web service APIs, REST APIs, and data sources.</td>
</tr>
<tr>
<td>Reporting</td>
<td>Grants the developer access to reports and scheduled reports.</td>
</tr>
<tr>
<td>Workflow</td>
<td>Grants the developer access to the Workflow Editor and Activity Creator.</td>
</tr>
<tr>
<td>Service Catalog</td>
<td>Grants the developer access to catalog related file types such as catalog items, record producers, and variables.</td>
</tr>
<tr>
<td>Flow Designer</td>
<td>Grants the developer access to the Flow Designer design environment to create flows and actions. Script action steps require the <strong>Allow Scripting</strong> permission.</td>
</tr>
<tr>
<td>Service Portal</td>
<td>Grants the developer access to Service Portal editors and tools.</td>
</tr>
<tr>
<td>Tables &amp; Forms</td>
<td>Grants the developer access to model and layout related file types such as table columns, form layout, and list layout.</td>
</tr>
<tr>
<td>Manage ACLs &amp; Roles</td>
<td>Grants the developer access to security-related file types such as access controls and user roles.</td>
</tr>
<tr>
<td>Allow Scripting</td>
<td>Grants the developer write access to script fields such as those in business rules, client scripts, and Flow Designer script action steps.</td>
</tr>
</tbody>
</table>

7. Click **Save**. The system creates and assigns the developer roles to the application.

**Remove a developer**

Removing a user as a developer prevents the user from creating or changing content for the application.
Role required: admin or application administrator

If Application administration is enabled, only an application administrator of the target application can delegate developers to an application. If application administration is not enabled, an admin user can delegate developers.

1. Navigate to System Applications > Applications.
2. Click the application name of the application from which you are removing developers. The system opens the application record.
3. Click Manage Developers.
The system displays the Developer Permissions window.
4. Point to the developer you want to remove.
The system displays a minus icon next to the developer name.

5. Click the minus icon next to the developer name.
The system removes the developer and any associated application roles.

**Developer permissions**

Administrators can assign one or more developer permissions to a group or user.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All File Types</td>
<td>Grants the developer access to all application file types including some not granted by the other options.</td>
</tr>
<tr>
<td>Integrations</td>
<td>Grants the developer access to web service APIs, REST APIs, and data sources.</td>
</tr>
<tr>
<td>Reporting</td>
<td>Grants the developer access to reports and scheduled reports.</td>
</tr>
<tr>
<td>Workflow</td>
<td>Grants the developer access to the Workflow Editor and Activity Creator.</td>
</tr>
<tr>
<td>Service Catalog</td>
<td>Grants the developer access to catalog related file types such as catalog items, record producers, and variables.</td>
</tr>
<tr>
<td>Permission</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Flow Designer</td>
<td>Grants the developer access to the Flow Designer design environment to create flows and actions. Script action steps require the Allow Scripting permission.</td>
</tr>
<tr>
<td>Service Portal</td>
<td>Grants the developer access to Service Portal editors and tools.</td>
</tr>
<tr>
<td>Tables &amp; Forms</td>
<td>Grants the developer access to model and layout related file types such as table columns, form layout, and list layout.</td>
</tr>
<tr>
<td>Manage ACLs &amp; Roles</td>
<td>Grants the developer access to security-related file types such as access controls and user roles.</td>
</tr>
<tr>
<td>Allow Scripting</td>
<td>Grants the developer write access to script fields such as those in business rules, client scripts, and Flow Designer script action steps.</td>
</tr>
</tbody>
</table>

**Domain separation and delegated development**

This is an overview of domain separation and the Delegated Development feature. Domain separation allows you to separate data, processes, and administrative tasks into logical groupings called domains. You can then control several aspects of this separation, including which users can see and access data.

**Overview**

**Support: Data only**

Domain separation in this application is supported at the Data only level, meaning it supports the data security model of separating visibility of data from one domain to another. To learn more, see Application support for domain separation.

**How domain separation works with Delegated Development**

Delegated Development is not a full application but rather, a feature in the Custom Business Applications suite, meaning it works alongside other features, including domain separation.

**Application management**

The system offers several ways to manage applications. You must have the admin role to perform these procedures.

**Application management actions**

<table>
<thead>
<tr>
<th>Action</th>
<th>Options</th>
<th>Use case</th>
</tr>
</thead>
</table>
| Install an application | • Install new application  
<p>|                    | • Install updates        | Application testing. |</p>
<table>
<thead>
<tr>
<th>Action</th>
<th>Options</th>
<th>Use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the current application</td>
<td>Select an application from the applications list</td>
<td>Edit an application.</td>
</tr>
<tr>
<td></td>
<td>Select an application from the application picker</td>
<td></td>
</tr>
<tr>
<td>Update the application record</td>
<td>Add tables</td>
<td>Add application logic.</td>
</tr>
<tr>
<td></td>
<td>Add roles</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add menus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add dependencies</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add Update Sets</td>
<td></td>
</tr>
<tr>
<td>Set the application state</td>
<td>Deactivate an application</td>
<td>Retire a legacy application.</td>
</tr>
<tr>
<td></td>
<td>Activate an application</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete an application</td>
<td></td>
</tr>
<tr>
<td>Share an application</td>
<td>Make available on another instance</td>
<td>Move an application from</td>
</tr>
<tr>
<td></td>
<td>Upload to the ServiceNow Store</td>
<td>development to testing.</td>
</tr>
<tr>
<td>Remove an application</td>
<td>Deactivate</td>
<td>Retire an obsolete or legacy</td>
</tr>
<tr>
<td></td>
<td>Delete</td>
<td>application.</td>
</tr>
<tr>
<td></td>
<td>Uninstall</td>
<td></td>
</tr>
<tr>
<td>Move application files from one global</td>
<td>None</td>
<td>Edit a legacy application.</td>
</tr>
<tr>
<td>application to another</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Install an application**

Install applications from the ServiceNow application repository or the ServiceNow Store.

Publish one or more applications to the application repository, or purchase an application from the ServiceNow Store.

Role required: admin, sn_appclient.app_client_company_installer (can only install applications that match the instance company), sn_appclient.app_client_user

The Applications module contains all applications that you purchased from the ServiceNow Store or published to the application repository. You must install an application from the Applications module to activate it in an instance.

1. Log in to the instance on which you want to install the application.
2. Navigate to System Applications > Applications.
3. Click Downloads.
4. Click the Install button for the application you want to install.

**Note:** You can only install one application at a time. The system disables the Install button for other applications during installation.

If an application is not listed, contact your application developer to publish the application.
The installation progress bar appears.

Install an update

Update an application that you purchased from the ServiceNow Store or published to the ServiceNow application repository.

Role required: admin, or a delegated developer role granting full access for this application.

1. Log in to the instance on which you want to check for updates.
2. Navigate to System Applications > Applications.
3. Click Updates.
4. Click the Update button for the applications you want to update.
   If an application is not listed, no updates are available for these applications.

Select an application from the applications list

Use the Applications list to open a custom application record.

When you edit an artifact from another application scope, the contextual development environment displays a warning message you can use to switch to that scope.

1. Navigate to System Applications > Applications.
2. Click the button for the application type you want to display.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop</td>
<td>Displays applications that have been created on this instance.</td>
</tr>
<tr>
<td>Downloads</td>
<td>Displays applications that have been downloaded on this instance.</td>
</tr>
<tr>
<td>Updates</td>
<td>Displays available updates for applications downloaded on this instance.</td>
</tr>
</tbody>
</table>
3. Click the application name or the Edit button for the application you want to work on.
   If your current application does not match this application, the system displays a warning message.

The Marketing Events application cannot be changed because Book Rooms is selected in your application picker. Switch to Marketing Events.

4. If needed, click the link to switch to the other application.

Select an application from the application picker

Use the application picker to select the application to which any new records and updates apply.

Application developers must select an application as their current scope context. Any new records become part of this application. The system also uses the current application scope to determine if the developer can view or make changes to records in other scope contexts.

1. Click the gear icon from the banner frame.
2. From **Application**, select the application where you want to save changes.

**Preserve unpublished applications during a system clone**

Application developers must manually save a copy of each application currently in development prior to cloning over their development instance.

- Role required: admin
- Write access to the application record
- A source control repository
The cloning process does not preserve version differences for applications in development. Instead, the system clone only copies the application version installed on the source instance onto the target instance. If the target instance had a development version of the same application, the application will be editable after the clone, but it will be at whatever version was installed on the source instance. If the application was missing from the source instance, the cloning process deletes the application from the target instance.

1. Use one of these actions to preserve the application on the clone target instance.

<table>
<thead>
<tr>
<th>Application version state</th>
<th>Action to take</th>
</tr>
</thead>
<tbody>
<tr>
<td>The application version on the clone target instance is different than the source instance version.</td>
<td>Export each application from the clone target instance. Choices include:</td>
</tr>
<tr>
<td></td>
<td>· (Recommended) Link each application to a source control repository.</td>
</tr>
<tr>
<td></td>
<td>· Publish each application to an update set.</td>
</tr>
<tr>
<td>The application is only available on the clone target instance.</td>
<td>None. The system clone process will copy this application version onto the target instance during the clone.</td>
</tr>
<tr>
<td>The application version on the clone target instance is the same as the source instance.</td>
<td></td>
</tr>
</tbody>
</table>

2. Request a system clone of the source instance over the target instance. For example, clone your production instance over your development instance.

3. After the clone process finishes, log in to the clone target instance.

4. If you saved each application to a source control repository, use one of these actions to retrieve them from the source control repository.

<table>
<thead>
<tr>
<th>Application installation state</th>
<th>Action to take on clone target</th>
</tr>
</thead>
<tbody>
<tr>
<td>The application was previously installed on the source instance.</td>
<td>Apply remote changes from source control repository.</td>
</tr>
<tr>
<td>The application was never installed on the source instance.</td>
<td>Import the application from source control repository.</td>
</tr>
</tbody>
</table>

5. If you saved each application to an update set, use one of these actions to retrieve them from the update set.

<table>
<thead>
<tr>
<th>Application installation state</th>
<th>Action to take on clone target</th>
</tr>
</thead>
<tbody>
<tr>
<td>The application was previously installed on the source instance.</td>
<td>1. Delete the application version cloned from the source instance. 2. Load the update set containing the current application version.</td>
</tr>
</tbody>
</table>
The applications previously in development are available for further development on the clone target instance.

**Preserve the Marketing Events application**

Suppose your company previously created version 1.0 of a custom application called Marketing Events. You have already published version 1.0 of the Marketing Events application to the application repository and installed it on your production instance.

Over time, users have submitted enhancement requests for the application, and you decide to develop version 2.0 of the Marketing Events application on a non-production instance to address these requests. As development nears completion, you want to update your non-production instance to the latest copy of production for some comprehensive testing.

Since you previously used a source control integration to develop version 1.0 of the Marketing Events application, you have already linked the Marketing Events application to a source control repository. You commit version 2.0 of the Marketing Events application to the source control repository.

You schedule a clone of the production instance over the development instance. After completion, you log in to the development instance and see that it has version 1.0 of the Marketing Events application, because that was the version installed on the source instance.

Since the application was already installed on the source instance, you apply remote changes from the source control repository to receive the latest application version. The development instance now has version 2.0 of the Marketing Events application available for further development and testing.

**Application sharing**

Administrators can share applications that are complete and are ready for use on other instances. Application developers can share applications using one of the following methods.

<table>
<thead>
<tr>
<th>Application sharing methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sharing method</strong></td>
</tr>
<tr>
<td>Publish to the application repository</td>
</tr>
<tr>
<td>Publish to the ServiceNow Store</td>
</tr>
<tr>
<td>Publish to an Update Set</td>
</tr>
<tr>
<td>Push to team development instances</td>
</tr>
</tbody>
</table>
Publish an application to the application repository

Publish a custom application to the application repository so that it can be installed on other instances in your organization.

To allow a developer to publish an application to the application repository, delegate the Publish to App Repo permission to the developer. For more information, see Add a developer.

Role required: admin, or delegated_developer with Publish To App Repo permission enabled

2. Open the In Development tab.
3. Open the application record that you want to publish to the application repository.
4. Click the Publish to My Application Repository related link.
5. Click Submit.

Install the application on company instances so that your organization can start using it. For more information, see Install an application from the application repository.

By default, after you publish an application to the application repository, all your company instances are entitled to the application automatically. To limit which company instances are entitled to the application, access the application repository by going to https://apprepo.servicenow.com, and then change the entitlement type for the application. For more information, see Manage application entitlements from the application repository.

Publish an application to the ServiceNow Store

Publishing an application to the ServiceNow Store makes it available to everyone.

To publish an application to the ServiceNow Store:

- Create an application within a private application scope.
- Join the Technology Partner Program.
- Have the application certified.

Note: You cannot publish global applications to the ServiceNow Store.

Role: admin

After you have met the prerequisites, you can publish the application to the ServiceNow Store.

1. Navigate to System Applications > Applications.
2. Click the Develop tab.
3. Open the application record you want to publish to the ServiceNow Store.
4. Click the Publish to Store related link.
5. Optional: Fill in the fields, as appropriate (see table).
6. Enter your HI credentials.
7. Click Submit.
   The system uploads the current version of the application to the ServiceNow Store allowing other users to download it.

Create application files to include sample data
Include sample records from an application data table when sharing a custom application.
The system can export selected records as application files that are included as part of the application update set when you share it. Including application files in an update set is not intended for the export and import of large numbers of records between instances. If you are trying to move data between instances, see Importing from another ServiceNow instance instead.

The application data only includes the version of the records that existed when the records were shared. The system does not update this snapshot of the application data when the records change. Application designers can include data on a table by table basis.

1. Navigate to the list for an application data table.
2. Filter the list to display the records you want to include.
3. Perform the appropriate action for the list version.

<table>
<thead>
<tr>
<th>Version</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>List v2</td>
<td>Open any column context menu and select Create Application Files.</td>
</tr>
<tr>
<td>List v3</td>
<td>Open the list title menu and select Create Application Files.</td>
</tr>
</tbody>
</table>

4. For Load When, select when the application record includes application data.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Install and Upgrades</td>
<td>Includes application data whenever the application is installed or upgraded.</td>
</tr>
<tr>
<td>New Install</td>
<td>Includes application data only when the application is installed.</td>
</tr>
<tr>
<td>New Install with Demo Data</td>
<td>Includes application data only when the application is installed with demo data.</td>
</tr>
</tbody>
</table>

5. Click OK.

The system adds the records to the application files related list.
6. Repeat steps 1–5 for each application data table you want to include.

Publish an application to an Update Set

Publishing an application creates an update set containing the current version of all application configuration records.

You can use this update set as a backup file for auditing purposes or to transfer the application to another instance.

1. Navigate to System Applications > My Company's Applications.
2. Click the Develop tab.
3. Open the application record you want to create an update set for.
4. Click the Publish to Update Set related link.
5. Optional: Fill in the fields, as appropriate (see table).
Publish to Update Set

Publishing an application lets you transfer it to another ServiceNow instance

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application name</td>
<td>(Read-only) Displays the name of the application that you are publishing.</td>
</tr>
<tr>
<td>Version</td>
<td>Enter version information to append to the Update Set name in dot-notation such as 1.2.3. The platform saves the information you enter here in the application Version field. The Update Set has the name &lt;Application name&gt; - &lt;Version&gt;. If you leave this field blank, the initial Update Set has the name &lt;Application name&gt; and subsequent Update Sets have the name &lt;Application name&gt; - &lt;Sequential number&gt;.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter a description for the Update Set. By default, this field contains the short description of the application.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Include data</td>
<td>Select the check box to include a limited number of data records from each table in the application. Use this feature to package sample data with your applications.</td>
</tr>
</tbody>
</table>

**Warning:**

Using this feature to migrate large quantities of data records between instances can cause performance issues, as it is not intended for this purpose. To migrate data, use an instance-to-instance import. You can adjust the maximum number of data records to include with an application.

See *Import Sets key concepts*.

**Note:**

- If your sample data includes tables with record numbering, the current counter value is also transferred. When the update set is applied on another instance, the counter is set to the larger of the sample data or the target instance counter.
- For translated fields, only records in English are transferred.

6. Click **Publish**.

   A new update set is created and the latest update of each application file in the application is copied into it. The update set is marked as complete.

7. Transfer the update set to another instance according to your test process.
   - Retrieve the update set from the source instance.
   - Save the update set as an XML file.

8. Run any fix scripts that are included in the application.

**Application removal**

Administrators have several options for removing unwanted applications.

**Delete an application**

You can delete applications that are no longer needed.

Typically, you only delete applications that have never been shared with other instances. This process ensures you can support any published applications.
By default, deleting an application also deletes all the records that are associated with the application, such as tables, business rules, and menus.

1. If you have need to reinstall the application at a later time, publish a backup of your application to the ServiceNow application repository, ServiceNow Store, or an update set. See Application sharing.
2. Navigate to System Applications > Applications.
3. Click the button for the application type you want to update.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop</td>
<td>Displays applications that have been created on this instance.</td>
</tr>
<tr>
<td>Downloads</td>
<td>Displays applications that have been downloaded on this instance.</td>
</tr>
</tbody>
</table>

4. Open the application record.
5. Click Delete.

The system displays a confirmation dialog box indicating the number of application files in the application to be deleted.

6. Optional: Click Show files to cancel deletion and navigate to a list of the application files in the application.
7. Click OK.
8. In the confirmation dialog box, enter delete and click OK.

The system deletes the custom application record unless the application extends other applications.

Uninstall applications

Uninstall applications that are no longer needed on an instance.

Uninstalling an application removes all application files associated with the application. You can remove application data as well.
Note: You cannot uninstall applications on the Develop tab, but you can delete them. You also cannot uninstall ServiceNow-provided system applications.

1. Log in to the instance on which you want to remove an application.
2. Navigate to System Applications > Applications.
3. Click the name or icon of the installed application to uninstall.
4. Click the Uninstall related link.
   The Uninstall confirmation window appears. You can review the tables and records associated with this application before uninstalling.
5. Optional: Clear the Retain tables and data check box to delete all data associated with this application, such as request records. Leave this check box selected to remove only application files such as field labels, dictionary entries, and ACLs.
6. Click OK.
7. In the confirmation prompt that appears, enter uninstall and click OK.
8. After the application is uninstalled, click Done.

Move an application file between global applications

When you select an application in the application picker, new application files are automatically assigned to it. You cannot move an application file into or out of a private scope application.

1. Navigate to the application file in a list or form view. For example, navigate to System Definition > UI Policies.
2. Locate the Move to Application UI action.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lists</td>
<td>Select the check box beside each appropriate row, and then select Move to Application in the Actions choice list.</td>
</tr>
<tr>
<td>Forms</td>
<td>Right-click the header and select Move to Application.</td>
</tr>
</tbody>
</table>

3. Select the global application to which you want to move the records.
4. Click **Move**.
   The record and its descendant records are moved to the selected global application.

### Application tools

The platform provides several tools to create, manage, and deploy applications.

#### Automated Test Framework

With the Automated Test Framework (ATF), you create and run automated tests on your ServiceNow instance. When you upgrade or modify an instance, run these tests to confirm that the instance still works as designed.

- **Explore**
  - Automated Test Framework release notes
  - Upgrade to Kingston
  - Getting started with the Automated Test Framework
  - Domain separation in Automated Test Framework

- **Set up**
  - Platform feature - active by default

- **Administer**
  - Administering the Automated Test Framework
  - Administering REST test step configurations

- **Use**
  - Build and run your first automated test
  - Building and running automated tests with the Automated Test Framework
  - Passing data from one automated test step to another

- **Troubleshoot and get help**
  - Search the HI Knowledge Base for known error articles
  - Contact ServiceNow Technical Support

- **Develop**
  - Developer training
Getting started with the Automated Test Framework

If you are new to the Automated Test Framework, read this overview to learn what the framework can do. Next, follow the tutorial to create and run a test that uses the most basic of ATF features. After you feel comfortable with the basics, explore more advanced features provided by the ATF.

ATF features provide flexibility in how you test your instance:

- With a form test, you can create simple tests that mimic user actions with no scripting, such as
  - Open a form
  - Set field values
  - Validate field values and attributes (such as read-only)
  - Submit the form

- With a server-side scripting test, you can perform more complex operations, including
  - Perform unit tests using JavaScript, including tests using the Jasmine test framework.
  - Test business rules, script includes, and other scripts.
  - Create tests that operate on data that you define.

- With a REST test, you can create and send an Inbound REST request and verify the response.
  - Test any REST endpoint on the instance.
  - Use a REST request to create records as well as retrieve, update, or delete records created in a previous test step or that existed on the instance.
  - Verify the response status code, response headers, response time, and response payload.

- With a Service Catalog test, you can perform end-to-end testing for a catalog item.
  - Open a catalog item or a record producer.
  - Search for a catalog item.
  - Set variables values and catalog item quantity.
  - Validate variable states, values, and price.
  - Add item to a shopping cart.
  - Order a catalog item and submit a record producer.

- Many test steps return output variables whose values you can use as inputs to a later step. For example, you can use output variables to accomplish the following tasks:
  - Perform a server-side assert on a record you previously inserted.
  - Create a record as one user, then reopen its form as a different user.

- In addition to the steps built into the Automated Test Framework, you can create custom test step configurations. These custom steps can take input variables and return output variables that you define. You can only define custom test steps that run on the server. The Automated Test Framework does not support creating custom step configurations that run in the browser.

- The test framework automatically tracks and deletes any data created by running tests, automatically taking care of rolling back changes after testing.
Test suites enable you to execute a batch of tests in a specified order. In addition, test suites can be hierarchical, with suites nested within other suites. You can associate test suites with schedules that determine when the system runs the test suites.

Automated Test Framework basic concepts

Learn the basics of how the Automated Test Framework works.

When you create an automated test, you create a record in the Tests table. Each test record contains a related list of steps and each step defines an action for the test to execute (for example, Open a form for a new record on the incident table).

When you run a test, the framework creates a Test Results record that you can inspect to learn what happened.

Tests

The Automated Test Framework stores information about individual tests in the Tests table, with each record corresponding to a single test. Each test record has a related list of the steps that the test attempts to execute when you run it.
Example of a test record in the Automated Test Framework

<table>
<thead>
<tr>
<th>Test Steps</th>
<th>Add Test Step</th>
<th>Add Test Template</th>
<th>Search for text</th>
<th>Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Steps</td>
<td></td>
<td>Add Test Step</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display name</td>
<td>Description</td>
<td>Table</td>
<td>Execution order</td>
<td>Active</td>
</tr>
<tr>
<td>Test Step 1</td>
<td>Open a New Form</td>
<td>User [sys_user]</td>
<td>1</td>
<td>true</td>
</tr>
<tr>
<td>Test Step 2</td>
<td>Set Test Values</td>
<td>User [sys_user]</td>
<td>2</td>
<td>true</td>
</tr>
<tr>
<td>Test Step 3</td>
<td>Submit a Form</td>
<td>Omit</td>
<td>3</td>
<td>true</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
When you run a test, the framework creates a Test Results record. The Test Results records from a given test are available through a related list on the Test record.

For reference information about test records, see Test record form.

Test Step

The test steps related list defines the individual steps and the order in which the test should execute them. In an individual test step record, the Step config field defines the action to take and the Field values define the data needed to take that action. The Automated Test Framework comes with a default set of step types (or step configs), but advanced users can also define their own custom types.
Input variables

A step's field values are also known as input variables because they provide the input data the step needs to execute.

Test runners

If a test includes steps that involve a form or any other user-interface (UI) element, it runs those steps in a browser tab or window called a test runner or client test runner. If no test runner is available when you run a test, the system prompts you to open one. For details about working with test runners, see Working with client test runners.

What to do next

Now that you know a little about the Automated Test Framework, strengthen your understanding by trying a hands-on example. For step-by-step instructions, see Build and run your first automated test.

Build and run your first automated test

Follow these step-by-step instructions to create and run your first automated test. This test creates a new user record.

- Role required: admin
- The Automated Test Framework plugin must be activated. It is activated by default on zboot or upgrade.
- If necessary, enable test execution.

Create new test

Create a new automated test record.

1. Navigate to Automated Test Framework > Tests.
2. Click New.
   The system displays the Test new record form.
3. On the Test new record form, enter a name for your test in the Name field. The system will identify this test by this name whenever it displays a list of tests, for example, under the Tests module.

Note: The test execution property is disabled to prevent running tests on a production system. Run tests only on development, test, and other sub-production instances.
4. In the **Description** field, enter a description for your test.

5. Click **Save**.

The system creates a new test record and returns to the list of tests. For more information about creating new automated tests, see *Create new automated test*.

**Add the first step to the new test**

Add the first of three steps to the automated test.

1. If necessary, navigate to **Automated Test Framework > Tests**.
2. Click the row containing the test you want to contain the new test steps.
   The system displays the **Test** form.
3. On the **Test Steps** related list, click **Add Test Step**.
   The system displays the **Add Test Step** form.
4. In the middle column, click the row for the step type **Open a new form**, then click **Next**.
The system displays the **Add Test Step** form for the **Open a new form** step.

5. From the **Table** field, select the **User (sys_user)** table. Then click **Submit**.

![Add Test Step form](image)

The system creates the new step and returns to the test record.

6. Click **Update**.

The system creates a new test record and returns to the list of tests. For more information about adding steps to automated tests, see **Add steps to automated test**.

For some ideas on how to continue learning about the Automated Test Framework, see **Next steps with the Automated Test Framework**.
Add the second step to your automated test
Add the second of three steps to the automated test.

1. Click the row containing the test you want to contain the new test steps.

   The system displays the **Test** form.

2. On the **Test Steps** related list, click **Add Test Step**.

   The system displays the **Add Test Step** form.

3. In the middle column, click **Set Field Values**, then click **Next**.

   The system displays the **Add Test Step** form for the **Set values** step.

4. In the field values section, set **Last name** to **Test** and **First name** to **Otto** (or other names of your choice).

5. Click **Submit**.
Add the last of three steps to the automated test.

1. Click the row containing the test you want to contain the new test steps. The system displays the Test form.
2. On the Test Steps related list, click Add Test Step. The system displays the Add Test Step form.
3. In the middle column, click Submit form, then click Next. The system displays the Add Test Step form for the Submit a form step.
4. Leave all values as set by default and click Submit.

The system creates the new step and returns to the test record. The test record should now show the three steps you just added.
5. Click Update
The system returns to the list of tests. For more information about adding steps to automated tests, see *Add steps to automated test*.

**Run your test**

After adding test steps, run and monitor the progress of the automated test

1. If necessary to view the Tests list, Click **Tests**.
2. Click the row containing the test you just created.
   
   The system displays the **Test** form.
3. Click **Run Test**.
   
   Because this test includes a form step (any step involving a UI), the system displays a dialog box asking you to choose among any currently-running test clients or start a new test runner. To continue, select **start a new test runner** and click **Run Test**.

The system displays the Run Test progress dialog. For more information about running automated tests, see *Run automated test*.

**Monitor test progress and view test results**

Monitor the progress of the automated test and view its test results.

1. Monitor the progress of the test in the Run Test progress dialog. If needed, you can **cancel a test** even while it’s running.

2. **Note:** If your test creates data, the system rolls back that data after all steps in the test complete.

When complete, click **Go to Results** on the Run Test progress dialog to display the **Test Results** list, where you can **view and analyze the results**.

**Next steps with the Automated Test Framework**

After you feel comfortable creating and running simple tests, explore the more advanced features of the Automated Test Framework.

**Batch tests together with test suites**

If you commonly run a set of tests together, you can group them using test suites. To learn about test suites, see *Building and running automated test suites*.

**Pass data from one test step to another with input and output variables**

You can pass data from one test step to another using **input variables** and **output variables**. For more information, see *Passing data from one automated test step to another*.

**Reuse common sequences of steps with templates**

If you frequently add the same sequence of steps to your tests, save time by creating a template. To learn more about templates, see *Add a predefined list of steps (template) to an automated test*.

**Building and running automated tests with the Automated Test Framework**

How to perform the basic tasks in the Automated Test Framework.

To build and run any test, you always perform certain operations:

- **create the test**
• add steps to the test
• run the test
• view the test progress
• view the test results

Other operations you perform often, but not necessarily for every test. For tests involving form steps, you might view the results screenshots. For some tests, you might need to pass data from one step to another. Some steps frequently occur in the same sequence in many different tests, so you can add pre-defined sequences, called templates, to a test.

Create new automated test

Create a named automated test containing a series of steps to execute.

Role required: (atf_test_admin) or (atf_test_designer).

1. Navigate to Automated Test Framework > Tests.
2. Click New.
   The system displays the Test new record form.
3. On the Test new record form, enter a name for your test in the Name field. The system will identify this test by this name wherever it displays a list of tests, for example, under the Tests module.

4. In the Description field, enter a description for your test.
5. Click Save.
   The system creates a new test record and returns to the list of tests.

Add the steps for the new test.

Add steps to automated test

Create a series of steps for an automated test to run in a specified order.

Role required: (atf_test_admin) or (atf_test_designer).

You must have created the test to which you want to add steps.
This procedure adds steps one at a time. You can also add a batch of steps all at once by adding a test template.

1. If necessary, navigate to **Automated Test Framework > Tests**.
2. Click the row containing the test you want to contain the new test steps.
   The system displays the **Test** form.
3. On the **Test Steps** related list, click **Add Test Step**.
   The system displays the **Add Test Step** form.
4. In the left-most column, select whether the new step performs an operation related to a form or an operation executed on the server.
   In the middle column, the system displays the test steps available for the type you select in the left column.

5. **Note:** Start any sequence of steps that work with forms with the **Open a new form** or **Open an existing form** step. Close with **Submit form**.

   Click the row showing the step type you want.
6. If applicable, from the **Insert after** field, select the step you want to precede this step. If this is the first step in a test, the **Insert after** field does not appear.
7. Click **Next**.
   The system displays the **Add Test Step** form for the step you have chosen.
8. From the **Table** field, choose the table you want to test in this step.
9. Optional: In the **Execution Order** field, enter an integer representing the order in which you want this step to execute. For more information on **Execution Order**, see **Edit automated test step order**.
10. Fill in the fields that apply to this type of step.
    Some steps return output values that you can pass to a subsequent step's inputs. For more information, see **Pass values from one automated test step to another**.
11. Click **Submit**.
    The system creates the new step and returns to the test record.
12. Repeat Steps 3 - 11 to add additional steps for this test.

**Run the test.**

**Run automated test**

After creating an automated test, run it on a non-production instance.

Role required: admin, atf_test_admin or atf_test_designer.

You must have created the test you want to run.

The **test execution property** must be enabled. You must have an admin or atf_test_admin role to do so.

**Note:** The test execution property is disabled by default to prevent running tests on a production system. Run tests only on development, test, and other sub-production instances.

1. Navigate to **Automated Test Framework > Tests**.
2. If necessary to view the Tests list, Click **Tests**.
3. Click the row containing the test you want to run.
   The system displays the **Test** form.
4. Click **Run Test**.
Note: If the test execution property is not enabled, the Run Test button does not display. In this case, see the annotation at the top of the form, and click the link to enable running tests.

5. If the test includes a form step (any step involving a UI), or other kinds of UI test steps, the Pick a browser dialog appears before executing the tests. Use it to choose among any currently-running test clients, or start a new runner. For more information, review Browser recommendations for all tests and suites.

If the test only includes server test steps, the system executes the tests without displaying the Pick a Browser dialog.

Monitor the progress of the test in the Run Test progress dialog. When complete, click Go to Results on the Run Test progress dialog to display the Test Results list, where you can view and analyze the results.

Note: If your test creates data, the system rolls back that data after all steps in the test complete.

Pick a browser

If the test or test suite you are running (or re-running, in the case of failed tests) contains steps that work with a form (any step involving a UI), or any other UI test step element (such as Automated Service Catalog test steps), the Pick a Browser dialog appears after clicking Run Test, Run Test Suite or Re-run failed tests.

Role required: admin, atf_test_admin or atf_test_designer.

You must have created the test you want to run.

The test execution property must be enabled. You must have an admin or atf_test_admin role to do so.

Note: The test execution property is disabled by default to prevent running tests on a production system. Run tests only on development, test, and other sub-production instances.

1. All registered client test runners that are currently active appear in the Pick a Browser dialog. Chose the browser in which the test or test suite should run (or re-rerun, in the case of failed tests).

   (Current session) indicates that the browser is currently running. See Working with client test runners.

2. Click Manage your test runners here as needed to view all client test runners registered for the current user.

   See Automated Test Framework Active Manual Test Runners module.

3. When client test runners are only available in other browsers for the current user, Start a new test runner appears. Click it if you want to open a client test runner in this browser session.

4. Click the appropriate button to run the test. The button that appears is dependent on the type of test you are running:
   a) Click Run Test if running a single test.
   b) Click Run Test Suite if running a test suite.
   c) Click Re-run failed tests if re-running failed tests.
The test, test suite or failed tests run in the selected browser or client test runner. The Progress viewer appears for monitoring of the progress of the test run.

View progress of running automated tests

When an automated test is running, view its progress in the Run Test progress dialog.
Role required: (atf_test_admin) or (atf_test_designer).

The system must be running a test.

When you execute a test or test suite, the system automatically displays the Run Test progress dialog. If you close this dialog, you can re-display it from either the results list or the results page for the currently-running test or test suite.

1. If necessary, navigate to the Test Results list or Test Results page for the currently-running test or test suite.
   - If the system is currently running a test suite, navigate to Automated Test Framework > Suite Results. If desired, you can click the row for the running test suite to view the Suite Results page for that test suite.
   - If the system is currently running a test that’s not part of a test suite, navigate to Automated Test Framework > Test Results. If desired, you can click the row for the running test to view the Test Results page for that test.

2. Display the Run Test dialog.
   - If you are currently viewing the Test Results page or the Suite Results page, click Show Progress under Related Links.
   - If you are currently viewing the Test Results list or Suite Results list, right-click the row for the running test or test suite, then click Show Progress.

Passing data from one automated test step to another

Some automated test steps create data that you can use as an input to a subsequent step. You can pass data from one test step to another using input variables and output variables.

The term input variables is another name for the field values associated with a step. These values are known as input variables because they provide the input the step needs to accomplish its task. For example, the Open Form step has three input variables: Table, Record, and View.
Example test step showing input variables

Some types of step also have output variables. These are the values that later steps in the same test can use as input. For example, the Record Insert step has an output variable called `Record` which contains the sys_id of the newly-created record.

**Note:** Input data can also be passed to an automated test case from an external source such as a .csv file.

The test step form does not indicate if a test step has output variables or not. However, you can easily tell if any output variables are available to provide a value to any given input variable. If you can map the value of one step’s output variable to the current step’s input variable, the system displays the mapping icon to the right of that input field. When you click the variable mapping icon, the system displays a tree giving you access to any available output variables from previous steps.
Example test step showing output variables

For step-by-step instructions on how to assign the value of an output variable to another step's input variable, see *Pass values from one automated test step to another*.

For a list of test step configs that return one or more output variables and the definitions of those output variables, see *List of test steps with output variables*.

For an example of a test that passes variables using input and output variables, see *Automated Test Framework use case: reference a value from a previous step*. 

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Pass values from one automated test step to another
Assign a form field the value of an output variable returned from a previous step.

Role required: (atf_test_admin). You must have a previous test step that returns an appropriate output variable.

1. To the right of the field whose value you want to assign, click the input value icon.

The input value mapping control lists previous steps that create output variables. If no previous steps create output variables, the control displays the message: There are no elements to show.

2. Click the row for the step that contains the output variable you want to use as an input.

3. Click the output variable you want to use.
   If the output variable is an id for a glide record, the control displays a tree picker providing access to fields for this record.

4. Navigate through the tree picker hierarchy until you find and select the value you want.

List of test steps with output variables
Some automated test steps create data that you can use as an input to a subsequent step.

Test steps with output fields

<table>
<thead>
<tr>
<th>Test Step</th>
<th>Output variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click a UI Action</td>
<td>table</td>
<td>The table for the form that contains this UI action.</td>
</tr>
<tr>
<td></td>
<td>record</td>
<td>The sys_id of the record on which the action was clicked.</td>
</tr>
<tr>
<td>Submit a form</td>
<td>table</td>
<td>The table for the submitted record</td>
</tr>
<tr>
<td></td>
<td>record</td>
<td>The sys_id of the submitted record</td>
</tr>
<tr>
<td>Impersonate</td>
<td>user</td>
<td>The user id of the user impersonated.</td>
</tr>
<tr>
<td>Record Insert</td>
<td>table</td>
<td>The table to which the new record belongs.</td>
</tr>
<tr>
<td></td>
<td>record</td>
<td>The sys_id of the new record.</td>
</tr>
<tr>
<td>Record Query</td>
<td>table</td>
<td>The table queried</td>
</tr>
<tr>
<td></td>
<td>first_record</td>
<td>The first record resulting from the query</td>
</tr>
<tr>
<td>Replay Request Item</td>
<td>table</td>
<td>The table to which the replayed request item belongs.</td>
</tr>
<tr>
<td></td>
<td>request</td>
<td>The replayed request item.</td>
</tr>
</tbody>
</table>

Add a predefined list of steps (template) to an automated test

With test templates you can add a predefined list of steps to a test. Any list of steps that follows a set pattern makes a good candidate for a template.

Role required: (atf_test_admin) or (atf_test_designer). You must have created the test to which you want to add steps.
Many tests follow similar patterns. One common pattern, for example, is to open a form, set some field values, validate some field values, click a UI action, and submit the current form. If a template exists containing these steps, you can add them to a test all at once. The Automated Test Framework comes with default templates. You can also create custom test templates.

1. Navigate to Automated Test Framework > Tests.
2. Click the row for the test to which you want to add steps.
   The system displays the Test form.
3. On the Test Steps related list, click Add Test Template.
   The system displays the Add a test template dialog.
4. From the Table field, select the table you want to test with these steps.
5. From the Template field, select the template containing the steps you want to add.
6. Click Add.
   The system adds the template steps to the test. It also adds to the test description a set of instructions on how to complete the test from this template.
7. Following the instructions in the test description, edit each step added by the template to include the necessary information.

Proceed to edit or save the test as you normally would.

Building and running automated test suites

Create and run batches of tests with automated test suites.

You can group tests together in test suites. This enables you to run a group of related tests as one job.

Test suites can contain other tests suites as well as individual tests. In these cases, the upper-level test suite is called the parent and the test suite contained within the parent is called the child. A test suite that is a child of one test suite can also act as a parent to another test suite. You can nest test suites to build a multi-level hierarchy.

The figure below illustrates some important properties of hierarchical test suites. In this figure, Test Suite 1 is at the top-most level. All other tests and test suites are descendents of Test Suite 1. Individual tests are colored purple and test suites are colored light red. In this figure you can see several important features:

- Test suites can have children, individual tests cannot
- A test suite can be both a child and a parent. For example, Test Suite 1.3 is a child of Test Suite 1 and a parent of Test Suite 1.3.1 and Test Suite 1.3.2
- Test suites can have both individual tests and test suites as children.
- You can nest test suites to build a multi-level hierarchy.
Example of Test Suite Hierarchy

Create automated test suite

Group automated tests into a suite you can execute as a batch.

Role required: (atf_test_admin) or (atf_test_designer).

The tests you want to include in the test suite must already exist.

1. Navigate to Automated Test Framework > Test Suites.
2. Click New.
   The system displays the Test Suite New Record form.
3. In the Name field, enter a name for this suite.
4. Optional: In the Description field, enter a description for this test suite.
5. To include a test in the test suite, in the Test Suite Tests related list, click Insert a new row.. .
6. In the Test field, enter the name of the test to add to this test suite.
7. Optional: In the Order field for this row, enter a value to determine the order in which this test should execute within the test suite. By default, the system assigns a value to this field according to the order in which you add the tests.
8. In the **Abort on Failure** field for this row, enter a value that indicates whether you want the test suite to stop or continue if this individual test fails. By default, the system assigns the value `false` to this field. **False** means that if this test fails, the system still executes any further tests in the test suite.

9. Repeat steps 6 - 8 for every test you want to include in this test suite.

10. Click **Submit**.

### Add test to existing automated test suite

Add a test to a test suite that already exists.

Role required: (atf_test_admin) or (atf_test_designer).

The tests you want to include and the test suite must already exist.

1. Navigate to **Automated Test Framework > Test Suites**.
2. Click the row containing the test suite you want. The system displays the **Test Suite** form.
3. In the **Test Suite Tests** related list, click **Insert a new row**.
4. In the **Test** field, enter the name of the test to add to this test suite.
5. Optional: In the **Order** field for this row, enter a value to determine the order in which this test should execute within the test suite. By default, the system assigns a value to this field according to the order in which you add the tests.
6. In the **Abort on Failure** field for this row, enter a value that indicates whether you want the test suite to stop or continue if this individual test fails. By default, the system assigns the value `false` to this field. **False** means that if this test fails, the system still executes any further tests in the test suite.
7. Repeat steps 6 - 8 for every test you want to include in this test suite.
8. Click **Submit**.

### Add a child test suite to a parent test suite

Create or add to a multi-level test suite by including a child test suite within a parent test suite.

Role required: (atf_test_admin) or (atf_test_designer).

The parent test suite must already exist.

1. Navigate to **Automated Test Framework > Test Suites**.
2. If the child test suite already exists, open the child test suite form for editing. If the child test suite does not yet exist, create it. In both cases, leave the test suite form open.
3. In the **Parent Suite** field, enter the name of the test suite you want to act as the parent to this child.
4. If desired, add one or more tests to the child test suite.
5. Click **Submit**.

### Run automated test suite

After creating an automated test suite, run it on a non-production instance.

Role required: admin, atf_test_admin or atf_test_designer.

You must have created the test suite you want to run.

The **test execution property** must be enabled. You must have an admin or atf_test_admin role to do so.
This procedure outlines how to start a test suite manually. You can also schedule test suites to run at a later time. For more information, see Working with scheduled test suites.

1. Navigate to Automated Test Framework > Test Suites.
2. If necessary to view the Test Suites list, click Test Suites.
3. Click the row containing the test suite you want to run.
The system displays the Test Suite form.
4. Click Run Test Suite.

Note: If the test execution property is not enabled, the Run Suite button does not display. In this case, see the annotation at the top of the form, and click the link to enable running tests.

5. If the tests associated with the test suite include a form step (any step involving a UI), or other kinds of UI test steps, the Pick a Browser dialog appears before executing the tests. Use it to choose among any currently-running test clients, or start a new runner. For more information, review Browser recommendations for all tests and suites.

If the tests associated with the test suite only include server test steps, the system executes the tests without displaying the Pick a Browser dialog.

Monitor the progress of the tests. When complete, click Go to results on the progress dialog window to display the Test Results list, where you can view and analyze the results.

Schedule automated test suite

You can schedule one or more test suites to run at a specific date and time.

Role required: aff_test_admin or aff_test_designer.

You must have created the test suites you want to schedule.

To schedule a test suite, you need three components:

- a test suite record
- a schedule record specifying when you want the system to run the test suite
- a scheduled suite run record that associates the test suite to run with the schedule for running it

For more information about the capabilities of and requirements for scheduled test suites, see Working with scheduled test suites.

1. Navigate to Automated Test Framework > Schedules.
The system displays the list of existing test suite schedules.
2. To create a new schedule, click New.
The system displays the Suite Schedule record form.
3. On the Suite Schedule record form, enter the name of the schedule, the frequency with which to run associated suites, the time at which to run associated suites, and the timezone for this schedule.
4. Optional: To specify a condition that must be met for running associated test suites, check Conditional, then fill in the Condition text box with the appropriate script.
5. To add a test suite to run, navigate to the Scheduled Suites related list, then click New.
The system displays Scheduled Suite Run record form.
6. On the Scheduled Suite Run record form enter the appropriate data.
a) Enter the test suite to run
b) If the suite contains UI steps, enter any client constraints you wish to apply (such as browser to use). For more information on client constraints, see Scheduled Suite Run.
c) Add to the record's watchlist users you want the system to inform (by email) when the scheduled suite has finished
d) Click Submit

7. To add more test suites to this schedule, repeat Steps 5 and 6.
8. Click Update.
9. If your suite includes any test steps that work with a form – or any other element on the client side – follow these steps to open a browser window for running the client portion of the scheduled tests.
   a) Review Browser recommendations and requirements for all tests and suites as well as those that apply only to scheduled suite runs.
   b) In the Navigator, right-click Scheduled Client Test Runner then click the option to open in a separate tab or window, as you prefer.
   c) Leave open the browser window that's running the client test runner and return to the browser window that contains the Navigator.

Run a scheduled test suite using a script

Execute a scheduled UI test suite immediately using a script without having to wait for the scheduled time. You can use this method while trying to automate the process of running a test. You must have created and scheduled the test suites you want to run. See Create automated test suite and Schedule automated test suite, for more information.

Role required: atf_test_admin or atf_test_designer

The following steps might not be in-line with your software configurations.
1. Spin up a virtual machine (VM) on an operating system with the necessary browsers.
2. Use Scheduled Client Test Runner to open a browser on the instance.
3. Call the scriptable method

```java
new sn_atf.ScheduledRunsExecutor().setScheduleSysId("SYS_ATF_SCHEDULE_SYS_ID_HERE").start();
```

To run a test only when the script is called, set the Run field to On Demand in the Schedule form.

View tests results, suite results, and screenshots

Test and test suite results show how long it took the system to execute a test, which steps failed, and can include screenshots of form steps. You can also view reports comparing different runs of the same test suites.

View automated test results

You have multiple options for navigating to the test results, depending on where you are in the user interface. For example, if the Run Test progress dialog is displayed, you can click Go to Result. The procedure described here enables you to view test results from any location in the user interface.

1. Navigate to Automated Test Framework > Test Results.
2. Click the row containing the test whose results you want to view. The system displays the Test Results form.

3. To see details about a specific step, in the Test Results Items related list, click the row containing the step you want to view. The system displays the Test Result Item form.

View results screenshots from automated test

If the test has a UI component, the system takes screenshots of the UI. View these screenshots to gain further insight into the test results.

Role required: (atf_test_admin) or (atf_test_designer).

You must have run the test whose results screenshots you want to view.

Note: For best results with screenshots, leave the browser zoom level set to 100%.

1. Navigate to Automated Test Framework > Test Results.
2. Click the row containing the test whose results you want to view. The system displays the Test Results form.
3. To see a screenshot taken during the test, find the screenshot you want in the attachments list. Screenshots are named with the word screenshot followed by the timestamp (always in UTC time) for when the system recorded the shot. You can match the screenshot to the test step by comparing the step and screenshot timestamps.

View automated test suite results

View results from an automated test suite.

Note: You have multiple options for navigating to the test suite results, depending on where you are in the user interface. For example, if the Run Test progress dialog is displayed, you can click Go to Result instead. The procedure described here enables you to view test suite results from any location in the user interface.

1. Navigate to Automated Test Framework > Suite Results.
2. Click the row containing the test whose results you want to view. The system displays the Test Suite Results record

Compare results and execution times for different automated test and suite results

You can compare execution times for different runs of an automated test or automated test suite. You can also compare results over time for a single automated test suite.

Role required: (atf_test_admin) or (atf_test_designer).

Compare execution times for different runs of an automated test

Compare how long it took the system to execute each test step across different runs of the same test.

1. Navigate to Automated Test Framework > Tests.
2. Select the row for the test whose results you want to compare. The system displays the Test record.
3. Navigate to the Test Results related list.
4. Check the rows for the test results you want to compare.
5. From the Actions on selected rows control, click Compare test step results. The system displays the Compare test result execution times bar graph.

Compare execution times for different runs of the same automated test suite

Compare how long it took the system to execute each test across different runs of the same test suite.

1. Navigate to Automated Test Framework > Suites.
2. Select the row for the test suite whose results you want to compare. The system displays the Test Suite record.
3. Navigate to the Test Suite Results related list.
4. Check the rows for the test suite results you want to compare.
5. From the Actions on selected rows control, click Compare test results. The system displays the Compare test result execution times bar graph.

Compare results for automated test suite runs (aging report)

Compare how many tests passed versus failed across different runs of the same test suite.

1. Navigate to Automated Test Framework > Suites.
2. Select the row for the test suite whose results you want to compare. The system displays the Test Suite record.
3. Under Related Links, click Display aging report. The system displays the Test suite aging report.

Re-run failed tests in an automated test suite

Re-run failed tests within a test suite without rerunning the entire suite.

Role required: admin, atf_test_admin or atf_test_designer.

The Re-run failed tests button appears on the suite result form and on the suite execution progress viewer after a suite with failed tests completes. It does not appear if test execution is disabled, the suite is deactivated, or the user does not have one of the required roles.

Re-run failed tests button actually re-runs all non-passing tests. These include canceled, skipped, failure, error tests, and suite results.

1. Navigate to either the suite result form or suite execution progress viewer for the completed suite that had failed tests.
2. Click Re-run failed tests.
3. If the re-run tests include a form step (any step involving a UI), or other kinds of UI test steps, the Pick a Browser dialog appears before executing the tests. Use it to choose among any currently-running test clients, or start a new runner. For more information, review Browser recommendations for all tests and suites.

If the re-run tests include only server test steps, the system executes the tests without displaying the Pick a Browser dialog.

The system re-runs the failed tests:

- The system creates a new suite result hierarchy for the re-run tests. The progress workers, test result, and test suite result records show the same suite hierarchy as the previous test suite. They do not include the tests or suites that passed in a previous run.
- If you delete or deactivate a child suite or test that failed, and then re-run it, the system does not execute that suite or test in the re-run.
• If you add a child suite or test to the suite to a failed test, and then re-run it, the system does not execute the added suite or test in the re-run.

To view the results from the previous run of a test or suite, click **Previous test result** on the **test result form** or **Previous suite result** on the test **suite result form**. These fields only appear for tests and suites that have been re-run.

**Editing automated tests**

Edit existing automated tests.

**Change automated test step**

If necessary, edit a test step after you create it.

Role required: (atf_test_admin) or (atf_test_designer).

1. Navigate to **Automated Test Framework > Tests**.
2. Click the row containing the test you want to edit.
3. On the **Test Steps** related list, click the row containing the step you want to edit.
   The system displays the **Set Field Values** form.
4. Edit the fields you want to change.
5. Click **Update**.

**Edit automated test step order**

By default, steps execute in the order in which you created them. You can change this order by editing the **Execution Order** field.

Role required: (atf_test_admin) or (atf_test_designer).

By default, the system assigns the value 1 to **Execution Order** for the first step created for a test. When you add a step, the system assigns it the next-highest available integer value. In other words, if the highest **Execution Order** for any step in the test has the value 7, the system assigns 8 to the new step. By editing these values, you can change the order in which the test executes the steps.

1. Navigate to **Automated Test Framework > Tests**.
2. Click the row containing the test you want to edit.
   The system displays the **Test** form.
3. In the **Test Steps** related list, edit the values in the **Execution order** column to determine the new order for the steps.
4. Click **Update**.

**Copy automated test**

Copy an existing test, which you can then re-name and modify.

Role required: (atf_test_admin) or (atf_test_designer).

1. Navigate to **Automated Test Framework > Tests**.
2. Click the row containing the test you want to copy.
   The Test record form opens.
3. Near the top-right of the Test record form, click **Copy Test**.
   An annotation displays confirming that the system has copied the test. After you dismiss this annotation, the system displays a new test record identical to the copied record, with the exception of the **Name**.
4. In the **Name** field, enter the name you want to assign to this new test.
5. Edit the test steps as desired and proceed as you would for any new test.
6. When you are finished making changes, click **Update**.

### Cancelling automated tests and test suites

You can cancel automated tests and automated test suites that are running or are queued to run.

How you cancel an automated test or automated test suite depends on whether the test or test suite is currently running or is queued to run.

#### Cancel queued automated test suite

You can cancel an automated test suite that is queued but has not yet run.

Role required: `(atf_test_admin)` or `(atf_test_designer)`.

1. Navigate to **Automated Test Framework > Run > Suite Run Queue**.
2. Select the action check boxes for the test suites to cancel.
3. From the action choice list, select **Delete**.

#### Cancel running and pending tests in an automated test suite

Cancel running and pending tests in a running suite test.

1. If necessary, **display the Run Test progress dialog**.
2. Click **Cancel Pending Steps**.
   The system displays a dialog asking you to confirm that you want to cancel this test execution.
3. In the Confirmation dialog, click **Yes**.

The system cancels any running and pending tests in this test suite and rolls back any data that tests in the test suite may have changed.

#### Cancel running automated test

You can cancel a running test from the Run Test progress dialog.

Role required: `(atf_test_admin)` or `(atf_test_designer)`.

The system must be running a test.

1. If necessary, **display the Run Test progress dialog**.
2. Click **Cancel Pending Steps**.
   The system displays a dialog asking you to confirm that you want to cancel this test execution.
3. In the Confirmation dialog, click **Yes**.

The system cancels the running test and rolls back any data changes the test made.

#### Cancel waiting automated test

You can cancel a waiting automated test from the Waiting/Running Test Runs module.

Role required: `(atf_test_admin)` or `(atf_test_designer)`.

1. Navigate to **Automated Test Framework > Run > Waiting/Running Test Runs**.
2. Select the action check boxes for the tests to cancel.
3. From the action choice list, select **Delete**.
Administering the Automated Test Framework

Enable or disable the Automated Test Framework, modify retention policies, move tests between instances, control user access to the Automated Test Framework, and create custom test step configurations and step environments.

For details about individual properties that control how the Automated Test Framework works, see Automated Test Framework Properties.

Creating custom test step configurations

Step configuration records (or step configs) define how each step type behaves. You can create new step configurations that define custom steps that run on the server.

When you add a step to an automated test, that step has a defined type such as Set Field Values, or Record Insert. Each step configuration has a number of characteristics that affect how steps of that type behave, including the inputs required, the actions performed, and so on.

These characteristics are defined in the step’s step configuration record.
Test step architecture

The *Step execution script* field determines the actions the system executes when a step with this config runs.

*Create custom step configuration*
Create a custom step configuration that can form the basis of new steps that run on the server.

Role required: (atf_test_admin)
The Automated Test Framework includes specific types of steps such as Open Form, Set Value, Assert Value and so forth. With the **Step Configurations** module, you can create steps that perform actions you specify. You can only create configurations for steps that run on the Server. You cannot create configurations for steps that run on the browser.

1. **Navigate to** Automated Test Framework > Administration > Step Configurations.
2. **Click New.**
   The system displays the Test Step Config form.
3. In the **Name** field, enter a name for your step type.
4. Leave **Active** checked.
5. Leave **Step environment** set to Server-Independent. You can only define step configurations that run on the server; you cannot define step configurations that run in the browser.
6. Optional: In the **Category** field, select the category to which you want to assign this step.
   Categories are used for filtering the step list in the Add Step dialog. For more information, see Category field example.
7. Optional: In the **Batch Order Constraint** field, choose one of the following values.
   - **None**: a step based on this configuration can appear at any point in a test
   - **Start Batch Execution**: if this test includes a batch with this step, this step must be the first in the batch
   - **Run in the middle of an execution**: if a test includes a batch with this step, this step must appear after the first and before the last step of the batch.
   - **Stop Execution**: if a test includes a batch with this step, this step must be the last step in the batch.
8. In the **Order** field, enter an integer specifying where steps with this configuration appear in the step list on the Add Test Step dialog. For more information, see the example using the Order field.
9. In the **Template reminder** field, enter the instructions you want to appear when this step is included in a test as part of a template. For more information, see the example of using the Template reminder field.
10. In the **HTML description** field, enter the text you want to appear when the cursor highlights this step on the Create New Step dialog. For more information, see the example using the HTML description field.

   **Note:** You may find it convenient to skip the next two steps, both of which involve writing scripts, until after you have added any input and output variables.

11. In the **Description generation script** field, add code to the provided template to generate the description assigned to a test step record when a step of this type is included in a test.
    For more information about writing this script, see Step description generation script. To see an example of where the system displays this description, see Description generation script example.
12. In the **Step Execution Script** field, add code to the provided template to define the script that executes when a step of this type runs. The script template provides instructions and examples for working with step inputs, outputs, and step results. For more details on the step execution script, see Step execution scripts.
13. **Click Submit.**
    The system creates a new test step configuration and returns to the list of test configurations.
14. Optional: To add input or output variables, re-open the step config record.
15. Optional: To add input variables to this step config, scroll to the Input Variables tab, then click **New**. Fill out the required fields for the new variable, then click **Submit**. Repeat until you have added all the input variables needed.
16. Optional: To control the order in which input variables appear on the New Step form, edit the values in the **Order** column for the Input Variables related list.
17. To add output variables to this step config, scroll to the Output Variables tab, then click **New**. Fill out the required fields for the new variable, then click **Submit**. Repeat until you have added all the output variables needed.

*Create a custom step configuration category*

Create a custom step config category.

Role required: *(atf_test_admin)*.

Categories are used for filtering the step list in the Add Step dialog. For more information, see *Category field example*.

1. Navigate to **Automated Test Framework > Administration > Step Configuration Categories**.
2. Click **New**. The system shows the **Test Step Config Category** form.
3. In the **Name** field, enter a name for your step category.
4. In the **Step Environment** field, enter the step environment in which steps under this category execute:
   - Server - Independent, if you want this category to contain steps that execute on the Server.
   - UI, if you want this category to contain steps that execute on browser.
   - Server - REST, if you want this category to contain steps that send Inbound REST messages to the instance.
5. In the **Display name** field, enter the category name you want to appear in the middle column of the Add Test Step dialog when this category is selected.

6. Click **Submit**.
Working with test step templates

Test step templates contain a list of steps to be added all at once to an automated test.

Create automated test steps template
Create a template containing a list of steps to be added all at once to an automated test.

Role required: (atf_test_admin).

Many tests follow similar patterns. One common pattern, for example, is to open a form, set some field values, validate some field values, click a UI action, and submit the current form. If a template exists containing these steps, you can add them to a test all at once. The Automated Test Framework comes with default templates out of the box. With this procedure, you create your own templates.

1. Navigate to Automated Test Framework > Administration > Test Templates. The system displays the Test Templates list.
2. Click New. The system displays the Test Template new record form.
3. In the Name field, enter a name for your template.
4. In the Test Template field, click the lock icon.

The Test Template field unlocks and expands to allow editing.
5. In the Test Template field, enter the name of the first test step to add to this template. The system adds the test step to the list.
6. Continue adding test steps — in the order in which you want them to appear — until you have added all the steps you want to include in the list.
7. In the Description field, enter a description of this template.
8. Click Submit.

Edit automated test steps template
Edit an existing test template.

Role required: (atf_test_admin).

1. Navigate to Automated Test Framework > Administration > Test Templates. The system displays the Test Templates list.
2. Click the row for the test template you want to edit. The system displays the Test Template form.
3. In the Test Template field, click the lock icon.

The Test Template field unlocks and expands to allow editing.
4. Note: You can delete steps anywhere in the list, but you can add steps only to the end of the list. You can re-order steps in a test after you add them using the template, but you cannot re-order steps in the template itself.

Add or delete steps to the template.

- To delete a step, select that step, then click the X icon.
- To add a step, in the Test Template field, enter the name of the step to add.
5. Continue adding and deleting test steps until it contains the steps you want.
6. Click Submit.

Enable or disable executing Automated Test Framework tests
Allow or prevent tests and test suites from executing on this instance.
Role required: (atf_test_admin).
The test execution property is disabled to prevent running tests on a production system. Run tests only on development, test, and other sub-production instances.
1. Navigate to Automated Test Framework > Administration > Properties.
2. Set the test execution property.
   - To enable test and test suite execution, check Enable test/test suite execution.
   - To disable test and test suite execution, uncheck Enable test/test suite execution.
3. Click Save.

Modify data retention policy for ATF test results
Modify the Auto Flush data retention policy, which designates how long the system retains data, and referencing data, for test and test suite results. You can change the frequency of flushing for the sys_atf_test_result or sys_atf_test_suite_result base tables. This setting controls how far back in time test result data is available.
Role required: (atf_test_admin).
The system regularly flushes data in the sys_atf_test_result and sys_atf_test_suite_result base tables, (and optionally, referencing data). By default, the system deletes test and test suite results data 30 days after creation. This task enables you to modify the Auto Flush retention policy for data stored in a specific base table (sys_atf_test_result or sys_atf_test_suite_result).
1. Navigate to Automated Test Framework > Administration > Table Cleanup.
   The system displays a list of the retention policies (Auto Flushes) it maintains for automated testing results tables.
2. Select the retention policy (sys_atf_test_result or sys_atf_test_suite_result) to modify.
   The system displays the record for this retention policy.
3. The Tablename field displays the name of the table to which the selected Auto Flush retention policy applies. Skip this field to accept the default.

   **Note:** Selecting another tablename in this field compromises the integrity of the Auto Flush record. Leave the tablename on existing ATF policies at the base system (default) value so it does not adversely affect ATF data retention behavior.

4. Specify how the system should determine the length of time for retention of data, and referencing data.
   a) In the Matchfield field, enter the field you want the system to use to monitor duration. For example, to specify that you want to delete data x amount of time after the system created it, leave Matchfield set to its default value of sys_created_on.
   b) In the Age in seconds field, enter the amount of time (in seconds) the system must wait before deleting the associated data and referencing data.
5. If you want to apply the policy to the specified data (for example, sys_atf_test_result), AND any data that references it, select Cascade delete (default value).
   Affected referencing data is stored the following tables: sys_atf_test_result_item, sys_atf_test_result_step, and sys_attachment (when table_name = sys_atf_test_result). If you
want the policy to simply flush data in the selected table (for example, sys_atf_test_result), and skip flushing of the referencing data, then clear **Cascade delete**.

6. In the **Conditions** field, specify the filter conditions to use for selection of data (and optionally, referencing data) for this Auto Flush retention policy.
   
   The default is **Retain indefinitely is false**, because the **Test Results record** also contains a **Retain indefinitely** check box that allows opting out of the auto flushes for specific test results.

7. Click **Update**.

**Manage status and retention policies for automated test client runners**

Modify how often active client test runners report in to the system and how long the system retains records for inactive client test runners.

Role required: `atf_test_admin`.

When you start a **client test runner**, the system registers that runner as **active**, meaning that it is either running a test or is available to run a test. While the runner is active, it reports in to the system at a specified interval. If the runner does not report in at the expected time, the system marks the runner as inactive. After a period of time the system deletes the runner. This task enables you to modify the **Automated Test Framework properties** that control these intervals.

1. Navigate to **Automated Test Framework > Administration > Properties**.
2. Navigate to the Test Runner Properties section.
3. Set the interval at which active client test runners report in to the system: in the **Test runner heartbeat interval** field, enter the reporting interval in seconds.
4. Set the period of time a test runner can remain inactive before the system deletes it: in the **Test runner timeout** field, enter the period of time in seconds.
5. Click **Save**.

**Control user permissions for Automated Test Framework**

Two user roles determine permissions for the Automated Test Framework: `atf_test_admin` and `atf_test_designer`.

Role required: `admin`

Set each Automated Test Framework’s user’s role to either `atf_test_admin` or `atf_test_designer`. For a list of permissions granted for each role, see **Automated Test Framework user roles**.

**Move automated tests from one instance to another**

Move automated tests from one instance to another using the normal process for update sets.

Role required: `(atf_test_admin)`.

You can move automated tests, automated test suites, and related data using update sets. For more information, see .

**Working with client test runners**

If an automated test includes steps that involve a form or any other user-interface (UI) element, it runs those steps in a browser tab or window called a **test runner** or **client test runner**. The Automated Test Framework supports two types of client test runners: Client Test Runners for manually started tests and Scheduled Client Test Runners for tests started by a schedule.

When test execution is enabled, clicking the Client Test Runner module opens the client test runner in the current browser session. If tests are waiting to be run, the Client Test Runner runs a waiting test. If no test is running, the client test runner displays the message **Waiting for a test to run**.
While the client test runner is idle, it checks every five seconds for waiting tests to start. This ensures that the system runs any tests it may have been unable to start because no client with the proper configuration was available.

**Note:** The client test runner monitors for tests from the current session and runs those tests as the logged-in user (unless it executes an Impersonate User step). If you start a client test runner, log out from the current session, then log in again, the client test runner will run using the new session.

When the client runner is active, it displays the activity of the currently running test in the **Execution Frame**.

**Note:** To prevent conflicts, the system allows only one test to run at a given time. This is true even if you have multiple client test runner windows open. If you submit tests to run when another test is already running, the system holds the new tests to run later. If a test remains waiting for more than ten minutes, the system cancels the test.

### Test execution property

To work with the client test runner module, the **test execution property** must be enabled.

**Note:** The test execution property is disabled to prevent running tests on a production system. Run tests only on development, test, and other sub-production instances.

If the test execution property is disabled when you select this module, the system displays a message and a link to the **automated test framework properties page** where you can enable it.

### Additional debugging functionality

If you have enabled **additional debugging functionality**, the client test runner module displays two tabs: **Execution Frame** and **Debug Info**. The **Execution Frame** displays the information normally shown by the client test runner and the **Debug Info** displays additional debugging information.

The system takes screenshots from the tests in the **Execution Frame** tab and records to the test result record.

### Browser recommendations for all tests and suites

- Some browsers have memory-management limitations that make it necessary to occasionally close and restart the browser when running the client test runner. These browsers include Internet Explorer, Edge, and older versions of Firefox. How often you should close the browser depends on the browser application’s memory allocation.
- Some browsers have features that throttle CPU time. To avoid problems, follow these guidelines:
  - run the client test runner in its own browser window
  - keep the client test runner at least partially visible on the screen
  - make certain the screen is not locked or shut off
- The client test runner takes screenshots as the tests run. For best results with screenshots, leave the browser zoom level set to 100%.
Browser recommendations for scheduled suites

The client test runners for scheduled suites have additional browser requirements.

- On OS X with the client test runner on Chrome or Safari: If the screen is locked or the client test runner tab is occluded when the system attempts to run the test suite, tests run significantly slower and may time out. For best performance, run client test runners for scheduled suites in a vm environment in which the screen does not become locked or disabled.
- The browser must meet the criteria you specified on the Scheduled Suite Run record.
- A client test runner meeting the criteria you specified on the Scheduled Suite Run record must be available to run the test suite at the scheduled time. The system cannot automatically open a client test-runner session.

Javascript window command intercepts

The Client Test Runner captures window object commands including console.log, console.error, alert, confirm, and prompt, with default responses where necessary.

- Any script that calls window.confirm receives a boolean response of true
- Any script that calls window.prompt receives the string response test value

Active Test Runners table

When you start a client test runner, the system registers that runner in the Active Test Runners table. You can view this table in the Active Manual Test Runners module and the Active Scheduled Test Runners module. These two modules provide views of the same table, filtered to show only manual or only scheduled test runners.

The Active Scheduled Test Runner module is useful when you create a scheduled suite run. For scheduled suite runs, you can specify the browser to use. To determine the name and version of a browser you want to use, start a scheduled test runner with that browser, then inspect that runner's record in the Active Scheduled Test Runners module.

The data in this table is transient. While the runner is active, it reports in to the system at a specified interval. If the runner does not report in at the expected time, the system marks the runner as inactive. After a period of time the system deletes the runner. You can modify these intervals on the Automated Test Framework properties page.

Administering REST test step configurations

Set request and response payload sizes, filter request and response headers, and create basic auth profiles.

Create a basic auth profile using the Automated Test Framework
Create basic auth profiles to specify basic authentication credentials for Send Request - Inbound test steps.

Role required: web_service_admin

The user name and password must be valid credentials on the instance where the tests using the profile are run.

1. Navigate to Automated Test Framework > Tests.
2. Select a test that uses a Send Request - Inbound step.
3. Select a Send Request - Inbound step.
4. In the **Basic authentication** field, select the hour glass to look up the available profiles.
5. On the **Basic Auth Configurations** form, select **New**.
6. In the **Name** field, enter a name for the profile.
7. In the **Username** field, enter a user name.
8. In the **Password** field, enter a password.
9. Click **Submit**.

**Filter REST request and response headers**
You can add a list of REST request and response headers that are not to be saved in step-result records. You can filter headers that might contain authentication credentials or other sensitive information. The phrase ‘Header redacted for security’ is saved instead.

To specify headers to be filtered, create a system property `glide.atf.rest.log.header_blacklist` with a comma-separated list of header names to be filtered. For information on adding properties, see ..

**Automated Test Framework REST properties**
These properties are installed with ATF REST.

**Note:** To open the System Property (sys_properties) table, enter `sys_properties.list` in the navigation filter.

<table>
<thead>
<tr>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>glide.atf.rest.log.header_blacklist</code></td>
<td>A list of headers whose content is not to be added to the log, or shown on a form. The phrase ‘Header redacted for security’ is saved instead.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Type:</strong> String</td>
</tr>
<tr>
<td></td>
<td>- <strong>Default value:</strong> empty</td>
</tr>
<tr>
<td></td>
<td>- <strong>Other possible values:</strong> A comma-separated list of header names.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Location:</strong> System Property (sys_properties) table</td>
</tr>
<tr>
<td><code>glide.atf.rest.request_payload_max_size</code></td>
<td>The maximum size of the request payload.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Type:</strong> String</td>
</tr>
<tr>
<td></td>
<td>- <strong>Default value:</strong> 100 Kb</td>
</tr>
<tr>
<td></td>
<td>- <strong>Maximum value:</strong> 1024 KB</td>
</tr>
<tr>
<td></td>
<td>- <strong>Location:</strong> System Property (sys_properties) table</td>
</tr>
<tr>
<td><code>glide.atf.rest.response_payload_max_size</code></td>
<td>The maximum size of the response payload.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Type:</strong> String</td>
</tr>
<tr>
<td></td>
<td>- <strong>Default value:</strong> 100 Kb</td>
</tr>
<tr>
<td></td>
<td>- <strong>Maximum value:</strong> 5120 KB</td>
</tr>
<tr>
<td></td>
<td>- <strong>Location:</strong> System Property (sys_properties) table</td>
</tr>
</tbody>
</table>
Optimizing automatic test performance

You can troubleshoot automatic test performance by inspecting system transaction log records and potentially shorten execution time by adjusting how often automatic tests capture screenshots.

**Managing automatic test screenshot settings**

Capturing a large number of screenshots can impair test performance. You can control which types of screenshots the system captures to minimize this effect.

By default, the system captures a screenshot every time it executes a form test step. This information can be useful for understanding test results, but may slow down how fast the system executes the test.

You can change automatic test framework settings so that the system captures all screenshots (as it does by default), no screenshots, or just screenshots for failed test steps.

You can change these settings to affect all tests run on this instance, or to affect just the current client test runner session. To affect all tests run on this instance, set the automatic test framework property from the automatic test framework properties page. To affect just the current client test runner session, set the screenshot mode from client test runner browser window.

Set the system property to control when the Automated Test Framework captures screenshots.

To control how often the current instance captures screenshots for form test-steps, set the screenshot capture mode on the automatic test framework properties page.

Role required: admin or atf_test_admin

Setting the screenshot mode from the automatic test framework properties page affects any new client test runners started on this instance. This setting does not affect currently-running test-runners.

1. Navigate to Automated Test Framework > Administration > Properties.
2. Set the enable or disable screenshot capture property. From the drop down menu
   - To capture screenshots for all steps, select Enable for all steps.
   - To capture screenshots only for failed steps, select Enable for all failed steps.
   - To capture no screenshots, select Disable for all steps.
3. Click Save.

Set the client test-runner property to control when the Automated Test Framework captures screenshots.

To control how often the current client test runner captures screenshots for form test-steps, set the screenshot capture mode on the client test runner browser window.

Role required: admin or atf_test_admin

Setting the screenshot mode from the client test-runner browser window affects only this client test-runner and persists until this test-runner session is closed. This setting does not affect any other running test-runners or any future test runners.

1. From the client test-runner browser window, click the form preferences icon
2. Click the Screenshot mode option.
   - To capture screenshots for all steps, select Enable for all steps.
   - To capture screenshots only for failed steps, select Enable for all failed steps.
   - To capture no screenshots, select Disable for all steps.
3. Click **Save**.

**View transaction data for automated test results**
To help troubleshoot performance issues with automatic tests, you can inspect related records from the system's transactions log (syslog_transactions) table.

Role required: admin or test_admin

**Note:** The system may not be able to log some transactions with an extremely-short duration.

Navigate to related transaction record in the system transaction log.
- From the **Steps Results record**, view transactions in the Step transactions related list.
- From the **Test Results record**, view transactions in the Test Transactions related list.

**Working with scheduled test suites**

You can schedule a test suite to run at a specified date and time.

To schedule a test suite, you need three components:
- a test suite record
- a schedule record specifying when you want the system to run the test suite
- a scheduled suite run record that associates the test suite to run with the schedule for running it

With this model, you can associate a schedule with many different test suites and vice-versa.

**Note:** You can schedule only test suites, not individual tests. Scheduled tests will only run if there is an open Scheduled Client Test Runner page matching the scheduled suite's browser conditions. Scheduled tests cannot run on a machine that is locked, powered down, or does not already have the browser open.

The watchlist on the test suite run record also allows you to specify users to receive an email when the system finishes executing the test suite run.

If the test suite contains one or more form steps (steps involving a user interface), you must ensure that a scheduled client test runner is actively running in a browser when the schedule triggers the suite run.

**Note:** See **Browser recommendations and requirements** for recommendations and requirements for running the client test runner.

For step-by-step instructions on how to schedule a test suite, see **Schedule automated test suite**.

**Designate users to receive email when system finishes running a scheduled test suite**

You can designate users to be notified when a scheduled test suite finishes executing.

Role required: atf-test-admin

1. Navigate to the scheduled test suite run record.
2. Add to the record's **watchlist** all users you want the system to notify when this scheduled test suite run completes.
Automated Test Framework scheduled test suite completed email
When the system completes executing a scheduled test suite, it sends an email to users on the Scheduled Suite Run record watchlist. This email contains information and links to further information about the Scheduled Suite Run and its results.

Suite Stats

The Suite Stats section of the email reports the number of suites and individual tests broken down by result status: Failed (F), Error (E), Skipped (S), Canceled (C), or Passed (P). For a description of what each status means, see Test Suite Results record.

Test Suite Results

The Test Suite Results section of the email reports test suite results over time. By default, this report includes only suites with failed tests, but you can change this setting with the Email properties field on the Automated Test Framework Properties page.

<table>
<thead>
<tr>
<th>Letter</th>
<th>Color</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>red</td>
<td>Test failed</td>
</tr>
<tr>
<td>P</td>
<td>green</td>
<td>Test passed</td>
</tr>
<tr>
<td>S</td>
<td>white</td>
<td>Test skipped</td>
</tr>
<tr>
<td>C</td>
<td>white</td>
<td>Test canceled</td>
</tr>
<tr>
<td>E</td>
<td>orange</td>
<td>Test has error</td>
</tr>
</tbody>
</table>

Each entry acts as a link to the result record for that run. If you point the mouse to any of these entries, the system displays the parent suite number, the parent suite end time, and the test result output.

Properties affecting email content

On the Automated Test Framework Properties form, you can set options affecting the format and content of the email.

Domain separation in Automated Test Framework

This is an overview of domain separation and the Automated Test Framework. Domain separation allows you to separate data, processes, and administrative tasks into logical groupings called domains. You can then control several aspects of this separation, including which users can see and access data.

Overview

Support: Data only

Domain separation in this application is supported at the Data only level, meaning it supports the data security model of separating visibility of data from one domain to another. To learn more, see Application support for domain separation.
Automated Test Framework use case examples

Use cases can help you construct tests for common scenarios.

Automated Test Framework use case: test basic form operations

This use case illustrates testing basic form operations with the Automated Test Framework.

Steps in test

1. Impersonate a user with the permissions needed to perform these steps, in this example ATF.User.
**Test step 1 - Impersonate**

1. Open a form, in this example a Catalog Task form.
3. On the open form, set field values, including for any mandatory fields. This example sets field values for **Assigned to**, **Short description**, and **Description**.
Test step 3 - Set Field Values

4. Submit the open form.
Test Step 4 - Submit a Form

Automated Test Framework use case: reference a value from a previous step

This use case illustrates assigning a form field the value of an output variable from a previous step. In this example, the second step references an output value from the first step.

Automated Test Framework: Pass values from one step to another example

1. Insert a record into the incident table. This example inserts a record into the Incident table.
### Step 1 - Record Insert

2. Open the record just inserted. Specify the record to open by assigning to the **Record** field, the output variable from Step 1.
3. Validate that fields on the open record have the values you expect.
Step 3 - Field Values Validation

Automated Test Framework use case: test a business rule

This use case illustrates testing a business rule with the Automated Test Framework.

This example tests a business rule that sets the value of **Locked out** to **true** when **active** is set to **false**.
Automated Test Framework: Business rule example

1. Impersonate a user with the necessary permissions. In this example, the step impersonates the admin user.
Step 1 - Impersonate

2. Open a form for the table to which this business rule applies. This example opens a new User form.
Step 2 - Open a New Form

3. Set values on the form that meet the requirements for submitting the form and for triggering the business rule. This example sets values for the **Active**, **Last name**, and **First name** fields.
Step 3 - Set Field Values

4. Submit the form.
Step 4 - Submit Form

5. Validate that the business rule ran. In this example the business rule tested sets **Locked out** to true if **Active** is set to false.
Step 5 - Field Values Validation

Automated Test Framework use case: test a data policy

This use case illustrates testing a data policy with the Automated Test Framework.

This example tests the data policy that sets the field **Assignment Group** to **mandatory** if **impact** is high.
Set Assignment group to mandatory if Impact is high

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Incident [incident]</td>
</tr>
<tr>
<td>Application</td>
<td>Global</td>
</tr>
<tr>
<td>Inherit</td>
<td></td>
</tr>
<tr>
<td>Reverse If False</td>
<td>Yes</td>
</tr>
<tr>
<td>Active</td>
<td>Yes</td>
</tr>
<tr>
<td>Short Description</td>
<td>Set Assignment group to mandatory if Impact is high</td>
</tr>
<tr>
<td>Description</td>
<td>Set Assignment group to mandatory if Impact is high</td>
</tr>
<tr>
<td>Conditions</td>
<td>Impact is 1 - High</td>
</tr>
</tbody>
</table>

### Related Links
- Convert this to UI Policy

### Data Policy Rules
- Data Policy = Set Assignment group to mandatory if Impact is high

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Mandatory</th>
<th>Read Only</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>assignment_group</td>
<td>True</td>
<td>Leave alone</td>
<td></td>
</tr>
</tbody>
</table>
### Automated Test Framework: Data policy example

1. Impersonate a user with the necessary permissions. In this example, the step impersonates an admin user.
Step 1 - Impersonate

2. Open a form for the table to which this data policy applies. This example opens a new incident form.
Step 2 - Open a New Form

3. Check that the data policy has not yet been triggered. In this example, the step checks to confirm that Assignment group is not mandatory.
### Step 3 - Field State Validation

4. If applicable, set the conditions that trigger the data policy. This example sets **Impact** to **High**.
Step 4 - Set Field Values

5. Validate that the data policy is enforced. In this example, the test confirms that the data policy set Assignment group to High after the previous step set Impact to High.
Step 5 - Field State Validation

Automated Test Framework use case: test a script include

This use case illustrates testing a script include with the Automated Test Framework.
To test a script include with the Automated Test Framework, create a test that performs these steps:

- a step that causes the system to execute the script include. Examples:
  - Run Server Side Script test step that calls the script include.
  - Form action step script – such as open a form, submit or form, or set a field value – that invokes the script include.

- A step that validates that the script include took the expected actions. The specific test step for this validation depends on what the script include is designed to do. Examples:
  - Field values validation, if the script sets a field value
  - Field state validation, if the script changes a field state
  - Record Query, if the script generates a new record

This example shows a test with one test step: Run Server Side Script. The script associated with this test step calls a script include that returns the value of its argument plus three. If the value returned from the script include is 8, the script include has worked as intended and the test passes.
### Automated test step for testing script include

```javascript
// assertEqual: A function used to compare that assertion.shouldBe == assertion.value;
// in case of failure it throws an Error and logs that the assertion by name has failed

// Example:
var testAssertion = {
    name: "my test assertion",
    should: "expected value",
    value: "actual value",
};
assertEqual(testAssertion); // throws Error, logs message to test step output

(function(outputs, steps, stepResult, describe, assertEqual) {
    var myNum = addThree(5);
    if (myNum)
        return true;
    else
        return false;
});
```
Script include to test with Automated Test Framework

Automated Test Framework use case: test a Service Catalog request

This use case illustrates testing a service catalog request with the Automated Test Framework.

With the Replay Request Item test step, you can test the service catalog ordering process once a request exists and has a record in the request item table. In the Kingston release, you cannot create an automated test for the process by which the user creates a new request.
1. Replay an existing service catalog request item. This test step inserts a new record in the `[sc_request]` table for the catalog request item RITM0010001.

<table>
<thead>
<tr>
<th>Display name</th>
<th>Description</th>
<th>Table</th>
<th>Execution order</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impersonate</td>
<td>Impersonate the user: Eric.Schroeder with user id: eric.schroeder</td>
<td>Approval [sysapproval_approvers]</td>
<td>2</td>
<td>true</td>
</tr>
<tr>
<td>Record Query</td>
<td>There should be at least one record in ‘sysapproval_approvers’ matching a query of Approval for = (empty) and Approver = Eric.Schroeder</td>
<td>Approval [sysapproval_approvers]</td>
<td>3</td>
<td>true</td>
</tr>
<tr>
<td>Record Update</td>
<td>Update a record into ‘sysapproval_approvers’ with the following values: Update the value of the field ‘State’ to ‘Approved’</td>
<td>Approval [sysapproval_approvers]</td>
<td>4</td>
<td>true</td>
</tr>
<tr>
<td>Impersonate</td>
<td>Impersonate the user: Natasha.Ingram with user id: natasha.ingram</td>
<td>Approval [sysapproval_approvers]</td>
<td>5</td>
<td>true</td>
</tr>
<tr>
<td>Record Query</td>
<td>There should be at least one record in ‘sysapproval_approvers’ matching a query of Approval for = (empty) and Approver = Natasha.ingram</td>
<td>Approval [sysapproval_approvers]</td>
<td>6</td>
<td>true</td>
</tr>
<tr>
<td>Record Update</td>
<td>Update a record into ‘sysapproval_approvers’ with the following values: Update the value of the field ‘State’ to ‘Approved’</td>
<td>Approval [sysapproval_approvers]</td>
<td>7</td>
<td>true</td>
</tr>
<tr>
<td>Impersonate</td>
<td>Impersonate the user: Bow.Ruggeri with user id: bow.ruggeri</td>
<td>Approval [sysapproval_approvers]</td>
<td>8</td>
<td>true</td>
</tr>
<tr>
<td>Record Query</td>
<td>There should be at least one record in ‘sysapproval_approvers’ matching a query of Approval for = (empty) and Approver = Bow.Ruggeri</td>
<td>Approval [sysapproval_approvers]</td>
<td>9</td>
<td>true</td>
</tr>
<tr>
<td>Record Update</td>
<td>Update a record into ‘sysapproval_approvers’ with the following values: Update the value of the field ‘State’ to ‘Approved’</td>
<td>Approval [sysapproval_approvers]</td>
<td>10</td>
<td>true</td>
</tr>
<tr>
<td>Impersonate</td>
<td>Impersonate the user: ATF.User with user id: ATF.User</td>
<td>Catalog Task [sc_task]</td>
<td>11</td>
<td>true</td>
</tr>
<tr>
<td>Record Query</td>
<td>There should be at least one record in ‘sc_task’ matching a query of Request Item = (empty)</td>
<td>Catalog Task [sc_task]</td>
<td>12</td>
<td>true</td>
</tr>
<tr>
<td>Record Update</td>
<td>Update a record into ‘sc_task’ with the following values: Update the value of the field ‘State’ to ‘Closed Complete’</td>
<td>Catalog Task [sc_task]</td>
<td>13</td>
<td>true</td>
</tr>
<tr>
<td>Record Query</td>
<td>There should be at least one record in ‘sc_task’ matching a query of Request Item = (empty) and State = Open</td>
<td>Catalog Task [sc_task]</td>
<td>14</td>
<td>true</td>
</tr>
<tr>
<td>Record Update</td>
<td>Update a record into ‘sc_task’ with the following values: Update the value of the field ‘State’ to ‘Closed Complete’</td>
<td>Catalog Task [sc_task]</td>
<td>15</td>
<td>true</td>
</tr>
<tr>
<td>Record Validation</td>
<td>Validate record from table ‘sc_req_item’ matches the following condition: State = Closed Complete</td>
<td>Requested Item [sc_req_item]</td>
<td>16</td>
<td>true</td>
</tr>
<tr>
<td>Record Validation</td>
<td>Validate record from table ‘sc_req’ matches the following condition: Request state = Closed Complete</td>
<td>Request [sc_request]</td>
<td>17</td>
<td>true</td>
</tr>
</tbody>
</table>
Service Catalog test step 1 details: Replay Request Item

This insertion triggers the Service Catalog Request workflow, which checks the price of the item, determines that it exceeds $1000.00, and therefore generates approval records for users belonging to the Catalog Request Approvals group. In this example, only one user—Eric Schroeder—belongs to this group.
2. Impersonate Eric Schroeder, the user who needs to approve this Service Catalog Request.

   ![Impersonate Eric Schroeder](image)

   **Test step - Impersonate Eric Schroeder**

3. Verify that the system created an approval record for Eric Schroeder and this request. Note that for the Approval for field, you assign the output value from Step 1: **Replay Request item > Request**.

   ![Record Query for Approval record](image)

   **Step 3 details: Record Query for Approval record**

4. Set the state of this approval record to **Approved**.
Step 4 details: set approval record to Approved

The Service Catalog Request workflow sees that all required approval records have the state of Approved and transitions to the Approval Action which marks the request record (sc_request) as Approved.
Step 4: Triggered workflow marks request record as approved

When the record in (sc_request) changes to the Approved state, an associated business rule generates request items (sc_request_item) for each item in the request. In this example, the request contains only one item, so the business rule inserts one record into the (sc_request_item) table. This insertion triggers the Service Catalog Item Request workflow.
Service Catalog Item workflow

The first activity in the Service Catalog Item Request workflow generates an approval record for the head of the department in which the requesting user works. In this example, the department head is Natasha Ingram.
Service Catalog Item workflow: step 4

5. The workflow does not continue until the department head approves it, so the next test step impersonates Natasha Ingram.
Step 5 - Impersonate User

6. Obtain the sys_id for the new approval record with the Record Query step. Note that Record Query creates an output variable with the sys_id of the first record returned from the query.

Step 6 - Record Query test step
7. Set the approval record to **Approved**.

**Step 7 - Approval User test step**

Note how Step 7 refers to the **First record** output variable from Step 6 to specify which record to approve. When the record is approved, the workflow transitions to the next Approval - User activity, which generates an approval record for the CIO. In this example, the CIO is Bow Ruggeri.
8. Impersonate Bow Ruggeri.
9. Obtain the sys_id for the approval record for Bow Ruggeri.
10. Set the approval record to Approved.

When the record is approved, the workflow transitions to the Approval Action activity which sets the record for this item in the (sc_request_item) table to Approved. The workflow transitions to the Catalog Task activity labelled Asset Mgmt. Fulfills Order. This Catalog Task activity generates a new record in the (sc_task) table that instructs a user in the Fulfillment group to order the item.
Step 10 - Service Catalog Item workflow

11. Impersonate a user in the Fulfillment group, in this example ATF.User.
Step 11 - Impersonate User test step

12. Obtain the sys_id for the new catalog task with the Record Query step. Note that Record Query creates an output variable with the sys_id of the first record returned from the query.

Step 12 - Record Query test step

13. Mark the (sc_task) record as Closed Complete.
Step 13 - Record Update test step

Note how Step 13 uses the First record output variable from Step 12 to specify which record to mark as Closed Complete.
When the record is marked Closed Complete, the workflow exits the Catalog Task activity along the In Stock exit path.
Step 13 - Service Catalog Item Request workflow

The workflow transitions to the Notification activity, then to the Catalog Task activity labelled Deploy Item to User. The Deploy Item to User Catalog Task activity inserts a new record into the (sc_task) table that instructs a user in the Deployment group to deliver the item.

14. Obtain the sys_id for the new catalog task with the Record Query step. Note that Record Query returns an output variable with the sys_id of the first record returned from the query.
**Step 14 - Record Query test step**

15. Mark the `{sc_task}` record as **Closed Complete**. Note how Step 15 uses the **First record** output variable from Step 14 to specify which record to mark as **Closed Complete**.
Step 15 - Record Update test step

When the record is marked Closed Complete, the workflow exits the Catalog Task activity, logs a message, and exits.
Step 15 - Service Catalog Item Request workflow

16. Verify that the request item in (sc_request_item) has the state *Closed Complete*. 
Step 15 details - Record Validation

17. Verify that the request (sc_request) has the state Closed Complete.
Step 16 - Record Validation test step

Automated Test Framework use case: Retrieve an incident using REST-Inbound

The **Get Newly Created Resource via REST API Test** test is provided with the Automated Test Framework, and uses the REST - Inbound and assert steps.

Review the **REST API, REST API Explorer**, and the **Send REST Request- Inbound- REST API Explorer** configuration step information before creating this test.

Role required: `atf_ws_designer`

This test creates an incident record, uses a REST- Inbound step to retrieve the record, and then uses assert steps to determine whether the request was successful.

You can use the Send REST Request- Inbound REST API Explorer step to build and test the request, or you can manually create the request using the Send REST Request- Inbound step. In either case, you must specify the basic authentication information on the Send REST Request- Inbound step form.

When creating your test, start with your REST API and determine what behavior you want to validate. You can then determine what test data to create. You can use other test step configuration categories to create, update, or delete records, and then use a Send REST Inbound step to retrieve, update, or delete the test data. Conversely, you can use the Send REST Inbound step to create records, and then use other test step configuration categories to validate that the records were created correctly.
This task steps you through creating the **Get Newly Created Resource via REST API Test** test.

1. Create a test.
   a) Navigate to Automated Test Framework > Tests, click New. The Test new record form is shown.
   b) Enter a test name and a description, and click Submit.
   c) Click the test created in the previous step. The Test form is shown.

2. Create test data.
   a) Click the Add Test Step button. The Add Test Step form is shown.
   b) Click a test configuration category, and then click a test configuration.
      To replicate the **Get Newly Created Resource via REST API Test** test, click Server > Record Insert, and then Next. The test configuration form you selected is shown.
   c) Fill in the information needed for the test configuration you selected.
      To replicate the **Get Newly Created Resource via REST API Test** test, on the Record Insert form, specify the incident table, and add a value for the Short description field, and click Submit.

3. Create the REST request.
   a) Click the Add Test Step button. The Add Test Step form is shown.
   b) Click a test configuration category, and then click a test configuration.
      To replicate the **Get Newly Created Resource via REST API Test** test, click REST > Send REST Request - Inbound REST API Explorer, and then Next. The REST API Explorer is shown.
   c) Fill in the information needed for the REST request, and click Send.
      To replicate the **Get Newly Created Resource via REST API Test** test, fill in the fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
<td>now</td>
</tr>
<tr>
<td>API Name</td>
<td>Table API</td>
</tr>
<tr>
<td>API Version</td>
<td>latest</td>
</tr>
<tr>
<td>tableName</td>
<td>incident</td>
</tr>
</tbody>
</table>

When you do not specify the **Basic authentication**, the REST API Explorer uses your credentials.

The **Create Automated Test Step** is shown after you click Send.

d) When ready, click **Create Automated Test Step**.
   The Send REST Request - Inbound test step is created.

e) Click the Send REST Request - Inbound step, and in the **Basic authentication** field, specify a basic auth configuration.
   If no configurations are available, you can create a basic auth configuration by clicking **New** on the **Basic Auth Configurations** form.
f) To replicate the Get Newly Created Resource via REST API Test test, in the Path field, click the contextual search button, and then click Record Insert > Record.

g) Click Update.

4. Create assert steps to verify the REST response.
   a) Click the Add Test Step button. The Add Test Step form is shown.
   b) Click a test configuration category, and then click a test configuration. To replicate the Get Newly Created Resource via REST API Test test, click REST > Assert Status Code, and then Next. The Assert Status Code form is shown.
   c) Enter the information needed for the form, and click Submit. To replicate the Get Newly Created Resource via REST API Test test, in the Operation field, select is and in the Status Code field enter 200. To replicate the Get Newly Created Resource via REST API Test test, repeat this step to create Assert Response JSON Payload is Valid, and Assert JSON Response Payload Element test steps.

List of Automated Test steps

The Automated Test Framework includes a set of specific types of test step. All test steps have inputs – the fields whose values you need to provide for the step to run. Some steps also have outputs – return values you can pass as input to a subsequent step.

UI test steps

UI test steps consist of Automated test form steps and Automated Service Catalog tests. Automated test form steps

Form test steps perform operations on the client (browser) side. These are operations that would normally be carried out by a human user.

Form test step: Open a new form

Opens a form to a new record in the specified table.

Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An Integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Description</td>
<td>(Read-only.) The system automatically populates this field from the field values you provide.</td>
</tr>
<tr>
<td>Table</td>
<td>Enter the name of the table for the new form.</td>
</tr>
<tr>
<td>View</td>
<td>Enter the name of the view in which you want this form to open. The testing user must have access to that view. If the name is not a valid form view, the form opens in its default view.</td>
</tr>
</tbody>
</table>

Form test step: Open an existing record
Opens a form to an existing record in the specified table.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Description</td>
<td>(Read-only.) The system automatically populates this field from the field values you provide.</td>
</tr>
<tr>
<td>Table</td>
<td>Enter the name of the table for the record you want to open.</td>
</tr>
<tr>
<td>Record</td>
<td>The record ID for the record you want to open.</td>
</tr>
<tr>
<td>View</td>
<td>Enter the name of the view in which you want this form to open. The testing user must have access to that view. If the name is not a valid form view, the form opens in its default view.</td>
</tr>
</tbody>
</table>

Form test step: Set Field Values
Sets fields on the current form to specified values.

To run this step, your test must have already opened a form using either the Open a New Form or Open an Existing Record step. In addition, your test cannot run this step after a Submit Form or Click a UI Action step.

The Field Values Validation, Set Field Values, Field State Validation, and UI Action Visibility steps can appear in any order.
Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Description</td>
<td>The system automatically populates this field from the field values you provide.</td>
</tr>
<tr>
<td>Table</td>
<td>The table for the form with the field to set.</td>
</tr>
<tr>
<td>Field values</td>
<td>The field to set.</td>
</tr>
<tr>
<td>Value</td>
<td>The value to which the system sets the field.</td>
</tr>
</tbody>
</table>

**Note:** Use the condition builder to set the field value. The condition builder displays an appropriate control for the field data type. For example, a reference field displays a **Lookup record** control.

Form test step: Field Values Validation

Validates field values on the current form.

To run this step, your test must have already opened a form using either the **Open a New Form** or **Open an Existing Record** step. In addition, your test cannot run this step after a **Submit Form** or **Click a UI Action** step.

The **Field Values Validation**, **Set Field Values**, **Field State Validation**, and **UI Action Visibility** steps can appear in any order.

For the Field Values Validation step, you specify the values you want to test using the standard conditions builder and can test several conditions against the same field. This step passes if the overall condition is satisfied and fails if it is not. If you need to test the values of individual fields independently of each other, include a separate Field Values Validation step for each value to test.

**Note:** The field values validation step works only with fields that belong to the record for the open form. For example, with the incident table, this step is not able to validate the Additional comments, Approval history, Comments, or Work notes "fields." The reason is that these are not actual fields on the incident record; they are UI controls that make it convenient to work with related tables. To validate these cases, use the Server test step, Record Validation, instead.
Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Timeout</td>
<td>The number of times the system attempts to execute the step, with a one-second delay between each attempt.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Table</td>
<td>The table for the form with the field to validate.</td>
</tr>
<tr>
<td>Conditions</td>
<td>The fields and values to validate. Includes only fields that are visible on the open form.</td>
</tr>
</tbody>
</table>

Form test step: Field State Validation
Validates the state of specified fields. States validated can include mandatory, non-mandatory, read-only, non-read-only, visible, and non-visible.

You can specify a maximum time to wait for the states of the fields to match the conditions in this step.

Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Timeout</td>
<td>The number of times the system attempts to execute the step, with a one-second delay between each attempt.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
</tbody>
</table>
### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Table</td>
<td>The table containing the fields to validate.</td>
</tr>
<tr>
<td>Visible</td>
<td>Check the lock icon to validate if fields on this form are visible. When checked, a slushbucket of available fields appear. To validate a field, move it to the <strong>Selected</strong> column.</td>
</tr>
<tr>
<td>Not visible</td>
<td>Check the lock icon to validate if fields on this form are not visible. When checked, a slushbucket of available fields appear. To validate a field, move it to the <strong>Selected</strong> column.</td>
</tr>
<tr>
<td>Read Only</td>
<td>Check the lock icon to validate if fields on this form are read-only. When checked, a slushbucket of available fields appear. To validate a field, move it to the <strong>Selected</strong> column.</td>
</tr>
<tr>
<td>Not read only</td>
<td>Check the lock icon to validate if fields on this form are not read-only. When checked, a slushbucket of available fields appear. To validate a field, move it to the <strong>Selected</strong> column.</td>
</tr>
<tr>
<td>Mandatory</td>
<td>Check the lock icon to validate if fields on this form are mandatory. When checked, a slushbucket of available fields appear. To validate a field, move it to the <strong>Selected</strong> column.</td>
</tr>
<tr>
<td>Not mandatory</td>
<td>Check the lock icon to validate if fields on this form are not mandatory. When checked, a slushbucket of available fields appear. To validate a field, move it to the <strong>Selected</strong> column.</td>
</tr>
</tbody>
</table>

### Form test step: UI Action Visibility

Verifies whether one or more UI Actions are visible or invisible on the current form.

The default visible UI Actions vary depending on the currently-impersonated user.

#### Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Execution Order| An integer specifying the order in which this step executes.  
As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values. |
| Active         | If this step is active true. Otherwise, false.                              |
Form test step: Click Modal Button
Clicks a button on a modal UI dialog window.

This step validates three conditions:
- the specified UI page was opened in a modal dialog
- the specified button is inside the modal dialog
- the specified button is visible and enabled

If any of the above validations fail, the step fails.

This step only succeeds if a modal dialog is currently open on the form, and if the specified button id exists on that modal dialog.

Include this step in your test after a Set Field Values step triggers a modal dialog that must be closed before performing more edits or assertions on the current form.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>The test step for this form.</td>
</tr>
<tr>
<td>Specify UI Page</td>
<td>The UI modal dialog to test.</td>
</tr>
</tbody>
</table>
Click button

The id attribute of the button element. For example, the OK button has an id of OK_button in the modal UI dialog window.

Form test step: Click a UI Action

Clicks a UI Action on the current form.

When this step runs, the system performs the action normally activated by that control. It also validates that the current form contains the control and that the control is visible and enabled. To run this step, your test must have already opened a form using either the Open a New Form or Open an Existing Record step. In addition, your test cannot run this step after a Submit Form or Click a UI Action step.

In the Kingston release, this step does not support UI actions of type Form context menu.

Note: Do not write tests that depend on the system displaying a specific page after executing Submit a Form or Click a UI Action. After these test steps, the system returns to the page that was open before the form was opened. The test cannot determine what that page was, so writing a test that expects a particular page can lead to unpredictable results.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Table</td>
<td>The table containing the UI Action to click.</td>
</tr>
<tr>
<td>UI Action</td>
<td>The UI Action to click.</td>
</tr>
<tr>
<td>Assert</td>
<td>Choose how submitting a form affects if this test passes or fails.</td>
</tr>
<tr>
<td></td>
<td>• Test passes only if form is submitted</td>
</tr>
<tr>
<td></td>
<td>• Test passes only if form submission is canceled</td>
</tr>
<tr>
<td></td>
<td>• Test passes in both of the above cases</td>
</tr>
</tbody>
</table>
Form test step: Submit a form
Submits the current form.

To run this step, your test must have already opened a form using either the Open a New Form or Open an Existing Record step. In addition, your test cannot run this step after a Submit Form or Click a UI Action step.

**Note:** Do not write tests that depend on the system displaying a specific page after executing Submit a Form or Click a UI Action. After these test steps, the system returns to the page that was open before the form was opened. The test cannot determine what that page was, so writing a test that expects a particular page can lead to unpredictable results.

### Outputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The table for the form that contains this UI action.</td>
</tr>
<tr>
<td>record</td>
<td>The sys_id of the record on which the action was clicked.</td>
</tr>
</tbody>
</table>

### Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Execution Order | An integer specifying the order in which this step executes.  
As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values. |
| Active        | If this step is active true. Otherwise, false.                                                                                             |
| Application   | The application scope in which the system runs this test or test suite.                                                                      |
| Test          | (Read only.) The test to which this step belongs.                                                                                         |
| Step config   | (Read only.) The test step for this form.                                                                                            |
| Assert Type   | Choose how submitting a form affects if this test passes or fails.  
- Test passes only if form is submitted  
- Test passes only if form submission is canceled  
- Test passes in both of the above cases |

### Outputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The table for the submitted record</td>
</tr>
</tbody>
</table>
Automated Service Catalog tests
If you have the catalog_administrator role, you can write automated tests to validate your catalog items in the Automated Test Framework (ATF). You can write end-to-end tests for both the requester and fulfiller flows.

The Automated Test Framework Service Catalog (com.glide.automated_testing_impl.service_catalog) plugin must be activated to support ATF for Service Catalog.

ATF for Service Catalog is supported only for single catalog item transactions. It is not supported for order guides and multiple items in the cart. For more information about ATF, see Automated Test Framework.

Service Catalog test step: Open a Catalog Item
Opens a catalog item.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>The sys_id of the submitted record</td>
</tr>
</tbody>
</table>

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the <strong>Execution Order</strong> values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td>(Read only) The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only) The current test step.</td>
</tr>
<tr>
<td>Description</td>
<td>(Read only) The system automatically populates this field from the field values you provide.</td>
</tr>
<tr>
<td>Catalog item</td>
<td>Select the catalog item that you want to open.</td>
</tr>
</tbody>
</table>

Service Catalog test step: Open a Record Producer
Opens a record producer.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>(Read only) The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only) The current test step.</td>
</tr>
<tr>
<td>Description</td>
<td>(Read only) The system automatically populates this field from the field values you provide.</td>
</tr>
<tr>
<td>Record Producer</td>
<td>Select the record producer that you want to open.</td>
</tr>
</tbody>
</table>

Service Catalog test step: Search for a Catalog Item
Searches for a catalog item or record producer in the specified catalog and category.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>(Read only) The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only) The current test step.</td>
</tr>
<tr>
<td>Description</td>
<td>(Read only) The system automatically populates this field from the field values you provide.</td>
</tr>
<tr>
<td>Search term</td>
<td>Term based on which you want to search for a catalog item.</td>
</tr>
<tr>
<td>Catalog</td>
<td>Catalog in which you want to search for the catalog item.</td>
</tr>
<tr>
<td>Category</td>
<td>Category in which you want to search for the catalog item.</td>
</tr>
</tbody>
</table>
Service Catalog test step: Set Variable Values

Sets variable values for the current catalog item or the record producer. For a catalog item, use this step after opening a catalog item page using the Open a Catalog Item step, and before using the Order Catalog Item step. For a record producer, use this step after opening a record producer page using the Open a Record Producer step, and before using the Submit Record Producer step.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Execution Order| An integer specifying the order in which this step executes.  
As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the **Execution Order** values. |
| Active         | If this step is active, **true**. Otherwise, **false**. |
| Application    | (Read only) The application scope in which the system will run this step. |
| Test           | (Read only) The test to which this step belongs. |
| Step config    | (Read only) The current test step. |
| Description    | (Read only) The system automatically populates this field from the field values you provide. |
| Item           | The current catalog item or record producer. |
| Variable values| Select the required variable to set its value. |

Service Catalog test step: Set Catalog Item Quantity

Sets the quantity for the current catalog item.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Execution Order| An integer specifying the order in which this step executes.  
As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the **Execution Order** values. |
<p>| Active         | If this step is active, <strong>true</strong>. Otherwise, <strong>false</strong>. |
| Application    | (Read only) The application scope in which the system will run this step. |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>(Read only) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only) The current test step.</td>
</tr>
<tr>
<td>Description</td>
<td>(Read only) The system automatically populates this field from the field values you provide.</td>
</tr>
<tr>
<td>Item</td>
<td>The current catalog item.</td>
</tr>
<tr>
<td>Quantity</td>
<td>Quantity of the catalog item.</td>
</tr>
</tbody>
</table>

Service Catalog test step: Validate Variable Values

Validates variable values on the current catalog item or record producer. For a catalog item, use this step after opening a catalog item page using the Open a Catalog Item step, and before using the Order Catalog Item step. For a record producer, use this step after opening a record producer page using the Open a Record Producer step, and before using the Submit Record Producer step.

Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, true. Otherwise, false.</td>
</tr>
<tr>
<td>Timeout</td>
<td>When specified, the Client Test Runner checks the validation conditions every second until the timeout value reaches or the conditions are met. If the timeout value exceeds or one of the validations is not correct, the step fails.</td>
</tr>
<tr>
<td>Application</td>
<td>(Read only) The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only) The current test step.</td>
</tr>
<tr>
<td>Item</td>
<td>The current catalog item or record producer.</td>
</tr>
<tr>
<td>Description</td>
<td>(Read only) The system automatically populates this field from the field values you provide.</td>
</tr>
<tr>
<td>Catalog conditions</td>
<td>Create conditions for variables. The test passes if the conditions evaluate to true.</td>
</tr>
</tbody>
</table>

Note: The label of a variable associated with a variable set reflects the variable set name. The format is variable_set_name » variable_name.
Validates state of the variables. Possible variable states are mandatory, not mandatory, read only, not read only, visible, and not visible.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Execution Order | An integer specifying the order in which this step executes.  
As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values. |
| Active      | If this step is active, **true**. Otherwise, **false**.                       |
| Timeout     | When specified, the Client Test Runner checks the validation conditions every second until the timeout value reaches or the conditions are met. If the timeout value exceeds or one of the validations is not correct, the step fails. |
| Application | (Read only) The application scope in which the system will run this step.      |
| Test        | (Read only) The test to which this step belongs.                             |
| Step config | (Read only) The current test step.                                           |
| Description | (Read only) The system automatically populates this field from the field values you provide. |
| Catalog Item | The current catalog item.                                                    |
| Visible     | To validate if a variable is visible, move it from the Available column to the Selected column. |
| Not visible | To validate if a variable is not visible, move it from the Available column to the Selected column. |
| Read only   | To validate if a variable is read only, move it from the Available column to the Selected column. |
| Not read only | To validate if a variable is not read only, move it from the Available column to the Selected column. |
| Mandatory   | To validate if a variable is mandatory, move it from the Available column to the Selected column. |
| Not mandatory | To validate if a variable is not mandatory, move it from the Available column to the Selected column. |

Service Catalog test step: Validate Price and Recurring Price  
Validates price and recurring price of a catalog item. Use this step after opening a catalog item page using the Open a Catalog Item step, and before using the Order Catalog Item step.
### Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td>(Read only) The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only) The current test step.</td>
</tr>
<tr>
<td>Description</td>
<td>(Read only) The system automatically populates this field from the field values you provide.</td>
</tr>
<tr>
<td>Catalog item</td>
<td>The current catalog item.</td>
</tr>
<tr>
<td>Price</td>
<td>Enter the price of the catalog item that should be validated.</td>
</tr>
<tr>
<td>Recurring price</td>
<td>Enter the recurring price of the catalog item that should be validated.</td>
</tr>
<tr>
<td>Recurring price frequency</td>
<td>Select the recurring price frequency of the catalog item that should be validated.</td>
</tr>
</tbody>
</table>

### Service Catalog test step: Add Item to Shopping Cart

Adds a catalog item to the shopping cart. Use this step after opening a catalog item page using the Open a Catalog Item step. After this step, you cannot use any other steps on the catalog item.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td>(Read only) The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only) The current test step.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Description</td>
<td>(Read only) The system automatically populates this field from the field values you provide.</td>
</tr>
</tbody>
</table>
| Assert type | Select how adding the catalog item to the shopping cart affects the test.  
  • Test passes only if the catalog item is successfully added to the shopping cart.  
  • Test passes only if the catalog item cannot be added to the shopping cart. |

Service Catalog test step: Order Catalog Item

Clicks **Order Now** for a catalog item. Use this step after opening a catalog item page using the Open a Catalog Item step. After this step, you cannot use any other steps on the catalog item. If the two-step checkout is false, a request is generated for the catalog item. If the two-step checkout is true, you are redirected to the cart preview page.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Execution Order | An integer specifying the order in which this step executes.  
  As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the **Execution Order** values. |
| Active | If this step is active, **true**. Otherwise, **false**. |
| Application | (Read only) The application scope in which the system will run this step. |
| Test | (Read only) The test to which this step belongs. |
| Step config | (Read only) The current test step. |
| Description | (Read only) The system automatically populates this field from the field values you provide. |
| Assert type | Select how ordering a catalog item affects the test.  
  • Test passes only if a catalog item is successfully ordered.  
  • Test passes only if a catalog item cannot be ordered. |

Service Catalog test step: Submit a Record Producer

Submits the current record producer. Use this step after opening the record producer page using the Open a Record producer step. After this step, you cannot use any other steps on the catalog item.
## Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the <strong>Execution Order</strong> values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td><em>(Read only)</em> The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td><em>(Read only)</em> The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td><em>(Read only)</em> The current test step.</td>
</tr>
<tr>
<td>Description</td>
<td><em>(Read only)</em> The system automatically populates this field from the field values you provide.</td>
</tr>
<tr>
<td>Assert type</td>
<td>Select how submitting a record producer affects the test.</td>
</tr>
<tr>
<td></td>
<td>• Test passes only if record producer is successfully submitted.</td>
</tr>
<tr>
<td></td>
<td>• Test passes only if record producer cannot be submitted.</td>
</tr>
</tbody>
</table>

### Server test steps

UI test steps consist of **Automated test server steps** and **Automated REST test steps**.

**Automated test server steps**

Server test steps perform operations on the server side. These are operations that would normally be carried out by an automated process such as a business rule.

Server test step: Impersonate

**Impersonate** specifies a user for executing subsequent steps in this test. It works for both server-side and browser-side steps and stays in effect until changed with another **Impersonate** step or until the test ends. The impersonation automatically ends when the test is over.

**Note:**

- Do not impersonate a user with the **test author** role. Doing so can lead to conflicts that interfere with executing the test.
- Do not rely on user IDs being consistent across different instances. The system dynamically assigns users IDs so the ID for a particular user often differs from one instance to the next.
- When exporting and importing automated tests, keep in mind that update sets do not update the user field.
Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>User</td>
<td>The user ID for the user to impersonate.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>The user id of the user impersonated.</td>
</tr>
</tbody>
</table>

Server test step: Record Query

Query the database to verify that a record exists matching the conditions set in this step.

Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Table</td>
<td>The table to query.</td>
</tr>
<tr>
<td>Conditions</td>
<td>The conditions with which to run the query.</td>
</tr>
</tbody>
</table>
Server test step: Record Insert
Inserts a record into a table with the field values you specify.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Enforce Security</td>
<td>Check to enforce ACLs.</td>
</tr>
<tr>
<td>Table</td>
<td>The table into which the record will be inserted.</td>
</tr>
<tr>
<td>Conditions</td>
<td>Specific field values to set when the test runs this step.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The table to which the new record belongs.</td>
</tr>
<tr>
<td>record</td>
<td>The sys_id of the new record.</td>
</tr>
</tbody>
</table>

**Note:** To ensure that the changes were applied, follow this step with a Record Validation step.
## Server test step: Record Delete

Deletes a specified record in a table.

### Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Enforce Security</td>
<td>Check to enforce ACLs.</td>
</tr>
<tr>
<td>Table</td>
<td>The table containing the record to be deleted.</td>
</tr>
<tr>
<td>Record</td>
<td>ID for the record to update.</td>
</tr>
<tr>
<td>Conditions</td>
<td>Specific field values to set when the test runs this step.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Server test step: Record Validation

Validates that a record meets the specified conditions on the server side.

For the Record Validation step, you specify the values you want to test using the standard conditions builder. You can apply several conditions to the same field.

This step passes if the overall condition is satisfied and fails if it is not. If you need to test the values of individual fields independently of each other, include a separate Record Validation step for each value to test.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Table</td>
<td>The table for the form with the field to validate.</td>
</tr>
<tr>
<td>Record</td>
<td>The record to test.</td>
</tr>
<tr>
<td>Conditions</td>
<td>Specific field values to validate when the test runs this step.</td>
</tr>
</tbody>
</table>

Server test step: Run Server Side Script

Executes a script on the server.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, \textit{true}. Otherwise, \textit{false}.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>The script to execute.</td>
<td>The javascript for the server to execute. Supports the Jasmine testing framework.</td>
</tr>
</tbody>
</table>

**Note:** If the script creates data, the system rolls back that data after all steps in the test finish.

Server test step: Replay Request Item
Get the item and requester from an existing request item, add the item to a new cart for that user, and place an order.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, \textit{true}. Otherwise, \textit{false}.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Original Request Item</td>
<td>The request item to replay.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The table to which the replayed request item belongs.</td>
</tr>
<tr>
<td>request</td>
<td>The replayed request item.</td>
</tr>
</tbody>
</table>

Server test step: Log
Logs a message and stores it as a step result.
The log message can contain variables and other information pertaining to the test. The message is stored as a step result.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Log</td>
<td>The message to log. To include the value of an output variable from a previous step, click input value icon( ) and follow the procedure to Pass values from one automated test step to another.</td>
</tr>
</tbody>
</table>

Server test step: Custom Scripted StepConfig

Provides an example of scripts for a custom step configuration.

This example checks if the user name provided starts with the letter A. This step is useful primarily to users with the (atf_test_admin) role. Users with the (atf_test_admin) role can view the example scripts by opening the record for this step in the Automated Test Framework Step Configurations module.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active, <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
</tbody>
</table>
### Automated REST test steps

You can create automated tests for Inbound REST APIs that you create. Creating tests for your custom REST APIs simplifies upgrade testing, and makes it possible to verify modifications to a REST API are backward compatible.

REST requests can only be sent to the current instance. You cannot send a request to another instance or third party/remote address.

The REST test configuration supports only XML and JSON response formats. Binary formats are not supported.

You can create tests that include steps from each of the test step configuration categories. The REST test configuration category contains the Send REST Request - Inbound and assert test configurations. Assert steps must immediately follow a Send REST Request - Inbound step. You can have multiple REST assert steps following a Send REST Request - Inbound step, but the assert steps cannot be separated from the Send REST Request - Inbound step by steps from other test categories.

### Authentication

As part of the Automated Test Framework, there are two situations when you send REST requests. The first is when you use the REST API Explorer to create and test a request. The second is when you run a test that contains a Send Request - Inbound step.

When you use the REST API Explorer to create and test a request, and the request requires authentication, the REST API Explorer uses your credentials. When the ATF runs the test, the credentials of the user who scheduled the test are used. This means that a test might fail unintentionally because of the difference in privileges between the user who created the test and the user that runs the test.

To address the issue of user credentials, you can create a basic-authentication profile for a test user and then on the Send Request - Inbound form, specify that the profile be used when the test is run.

REST test step: Send REST Request - Inbound - REST API Explorer

This test step begins with the REST API Explorer. Use the REST API Explorer to create and specify the HTTP method, path, query parameters, request headers, and body of a REST request, and then send the REST request to the current instance.
When you have tested the request, the Create Automated Test Step button is shown. Click Create Automated Test Step button to create the test step. This button does not appear until after the request has been sent. You cannot create a test step when the request payload is larger than the maximum request payload size property.

This test step creates the same test record as the Send REST Request - Inbound test step. After the test step is created, you cannot go back and use the REST API Explorer to update the test. All changes must be made on the Send REST Request - Inbound test step form.

No HTTP response validation is performed as part of this step. The step fails if the response payload size is too big, the request parameters are invalid, or the request could not be sent. Use the assert steps to validate the response.

You cannot use this step to send a request to another instance or third party/remote address.

These inputs are for the REST API Explorer. The fields you see depend upon the API selected. For more information on using REST APIs on your instance, see the REST API.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
<td>Lists the available namespaces for the request.</td>
</tr>
<tr>
<td>API Name</td>
<td>Lists the APIs available in the selected namespace.</td>
</tr>
<tr>
<td>API Version</td>
<td>Lists the versions of the API that are available. You can select the version available on your instance.</td>
</tr>
<tr>
<td>Path parameters</td>
<td>The part of the path that is after the API name. Path parameters are generally name value pairs where the allowable values are in a list.</td>
</tr>
<tr>
<td>Query parameters</td>
<td>Query parameters are added to the URI after the path. Query parameters are name value pairs. The REST API Explorer encodes the URI, so it is not necessary to encode query parameters and values.</td>
</tr>
<tr>
<td>Request headers</td>
<td>A list of name value pairs contained in the request header. The authentication header is set to Send as me to use the current user's credentials. To use the test step in production, you may need to change the Basic Authentication field in the Send REST - Inbound test step. Do not encode the name or value.</td>
</tr>
<tr>
<td>Request Body</td>
<td>The request content. Some requests do not have a body.</td>
</tr>
</tbody>
</table>

**REST test step: Send REST Request - Inbound**

Create a test step to send a REST request to the current instance. Specify the HTTP method, path, query parameters, request headers, and body of a REST request.

No HTTP response validation is performed as part of this step. The step fails if the response payload size is too big, the request parameters are invalid, or the request could not be sent. Use the assert steps to validate the response.

You cannot use this step to send a request to another instance or third party/remote address.

For more information on using REST APIs, see the REST API.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Basic authentication</td>
<td>If you are using a public API, you don’t need to provide authentication credentials.</td>
</tr>
<tr>
<td></td>
<td>If you are using a non-public API, you must provide the basic authentication information when doing the test. If you leave this field blank, you may receive unexpected results, for example, the response-status code might be 401 Unauthorized instead of 200 OK.</td>
</tr>
<tr>
<td></td>
<td>You must create or select a basic authentication profile to assign to test steps to avoid authentication issues when running the test.</td>
</tr>
<tr>
<td></td>
<td>Users must have the web_service_admin role to fully leverage the REST test steps. Without this role, users can not view or set basic authentication profiles needed for endpoints that require authentication. Without this role, users may still leverage the REST test steps for public APIs and unauthenticated access to non-public APIs. For full REST test step access, the atf_ws_designer role is available and contains the web_service_admin role along with the atf_test_designer role.</td>
</tr>
<tr>
<td>Method</td>
<td>Select the HTTP method to use.</td>
</tr>
<tr>
<td>Path</td>
<td>Enter the path. This field only accepts the portion of the URI after the instance. If you use <a href="https://dummy.com">https://dummy.com</a>, you get an error.</td>
</tr>
<tr>
<td>Query Parameters</td>
<td>Enter the parameter name and value. Do not encode the parameters.</td>
</tr>
<tr>
<td>Headers</td>
<td>Enter the header name and value. Do not encode the name or value.</td>
</tr>
</tbody>
</table>

**REST test step: Assert Status Code**

Assert that the HTTP response status code has the specified relationship to the specified value. You specify a numeric value of the status code and the relationship.
Assert steps must immediately follow a Send REST Request - Inbound step. You can have multiple REST assert steps following a Send REST Request - Inbound step, but the assert steps cannot be separated from the Send REST Request - Inbound step by steps from other test categories.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Operation</td>
<td>Select the relationship between the response code and the status code you specify.</td>
</tr>
<tr>
<td>Status code</td>
<td>Enter the value to test against the response code.</td>
</tr>
</tbody>
</table>

REST test step: Assert Status Code Name

Assert that the HTTP response status code name has the specified relationship to the specified value. You specify a value of the status code name, and the relationship.

Assert steps must immediately follow a Send REST Request - Inbound step. You can have multiple REST assert steps following a Send REST Request - Inbound step, but the assert steps cannot be separated from the Send REST Request - Inbound step by steps from other test categories.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
</tbody>
</table>
REST test step: Assert Response Time

Assert that the HTTP response time has the specified relationship to the specified value. You specify a value of the response time and the relationship.

Assert steps must immediately follow a Send REST Request - Inbound step. You can have multiple REST assert steps following a Send REST Request - Inbound step, but the assert steps cannot be separated from the Send REST Request - Inbound step by steps from other test categories.

Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Operation</td>
<td>Select the relationship between the response time value and the response time you specify.</td>
</tr>
<tr>
<td>Response time</td>
<td>Enter the time in milliseconds to compare against the response time.</td>
</tr>
</tbody>
</table>

REST test step: Assert Response Header

Assert the HTTP response header exists, or the header has the specified relationship to the specified value.

Assert steps must immediately follow a Send REST Request - Inbound step. You can have multiple REST assert steps following a Send REST Request - Inbound step, but the assert steps cannot be separated from the Send REST Request - Inbound step by steps from other test categories.
## Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td><em>(Read only.)</em> The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td><em>(Read only.)</em> The test step for this form.</td>
</tr>
<tr>
<td>Header</td>
<td>Enter the header name.</td>
</tr>
<tr>
<td>Operation</td>
<td>Select the relationship between the response header value and the response header you specify.</td>
</tr>
<tr>
<td>Value</td>
<td>The element value to use in the test. This field is not shown if the operation is <strong>is not empty</strong>.</td>
</tr>
</tbody>
</table>

### REST test step: Assert Response JSON Payload is Valid

Assert that the response payload is in valid JSON format.

Assert steps must immediately follow a Send REST Request - Inbound step. You can have multiple REST assert steps following a Send REST Request - Inbound step, but the assert steps cannot be separated from the Send REST Request - Inbound step by steps from other test categories.

### Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td><em>(Read only.)</em> The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td><em>(Read only.)</em> The test step for this form.</td>
</tr>
</tbody>
</table>

### REST test step: Assert Response XML Payload is Well-Formed

Assert that the response payload is well-formed XML.
Assert steps must immediately follow a Send REST Request - Inbound step. You can have multiple REST assert steps following a Send REST Request - Inbound step, but the assert steps cannot be separated from the Send REST Request - Inbound step by steps from other test categories.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
</tbody>
</table>

**REST test step: Assert XML Payload Element**

Assert the XML response payload element exists, or has the specified relationship to the specified value.

Assert steps must immediately follow a Send REST Request - Inbound step. You can have multiple REST assert steps following a Send REST Request - Inbound step, but the assert steps cannot be separated from the Send REST Request - Inbound step by steps from other test categories.

**Inputs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
</tbody>
</table>
### Field | Description
--- | ---
**Element path** | Specify the XML path to the element to be evaluated. For example `/result/short_description`

```json
{
   "result": {
       "number": "INC0020001",
       "short_description": "test"
   }
}
```

**Operation** | Select the relationship between the element in the response and the value you specify.

**Value** | The element value to use in the test. This field is not shown if the operation is **is not empty**.

---

REST test step: Assert JSON Response Payload Element

Assert the JSON response payload element exists, or has the specified relationship to the specified value.

Assert steps must immediately follow a Send REST Request - Inbound step. You can have multiple REST assert steps following a Send REST Request - Inbound step, but the assert steps cannot be separated from the Send REST Request - Inbound step by steps from other test categories.

### Inputs

| Field | Description |
--- | ---
**Execution Order** | An integer specifying the order in which this step executes. As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values. |

**Active** | If this step is active **true**. Otherwise **false**. |

**Application** | The application scope in which the system will run this step. |

**Test** | (Read only.) The test to which this step belongs. |

**Step config** | (Read only.) The test step for this form. |

**Element path** | Specify the SNC path to the element to be evaluated. For example `/result/short_description` for

```json
{
   "result": {
       "number": "INC0020001",
       "short_description": "test"
   }
}
```

See [Importing JSON files](#) for more information.
REST test step: Assert Response Payload

Assert the HTTP response payload has the specified relationship to the specified value. You specify the value and the relationship.

Assert steps must immediately follow a Send REST Request - Inbound step. You can have multiple REST assert steps following a Send REST Request - Inbound step, but the assert steps cannot be separated from the Send REST Request - Inbound step by steps from other test categories.

Note: The entire payload is used to look for a match. A large payload can affect performance.

Inputs

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Order</td>
<td>An integer specifying the order in which this step executes.</td>
</tr>
<tr>
<td></td>
<td>As you create steps, the system automatically assigns each step an incremental value. This causes the test to execute steps in the order in which you created them. You can change this default order by editing the Execution Order values.</td>
</tr>
<tr>
<td>Active</td>
<td>If this step is active true. Otherwise, false.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system will run this step.</td>
</tr>
<tr>
<td>Test</td>
<td>(Read only.) The test to which this step belongs.</td>
</tr>
<tr>
<td>Step config</td>
<td>(Read only.) The test step for this form.</td>
</tr>
<tr>
<td>Operation</td>
<td>Select the relationship between the response body and the value you specify.</td>
</tr>
<tr>
<td>Response body</td>
<td>The value of the response body to use in the test. Must contain the name and value to be compared as it appears in the response payload. The field must not contain any curly braces. This field is not shown if the operation is is not empty.</td>
</tr>
</tbody>
</table>

To check the short description in the response payload

```json
{"result":{"number":"INC0010040","short_description":"Test ATF Incident"}}
```
the **Response body** should contain

```
"short_description":"Test ATF Incident"
```

These formats are incorrect and the step fails.

- `{"short_description":"Test ATF Incident"}
- "{"short_description":"Test ATF Incident"}"
- `short_description: Test ATF Incident`
- `short_description:"Test ATF Incident"

## Automated Test Framework reference

Reference information for the Automated Test Framework.

### Automated Test Framework user roles

You can assign roles to define what specific users have permission to do with the Automated Test Framework.

**Automated Test Framework roles**

<table>
<thead>
<tr>
<th>Role Title (name)</th>
<th>Role Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>atf_test_admin</td>
<td>Has permission to:</td>
</tr>
<tr>
<td></td>
<td>• View the tests page</td>
</tr>
<tr>
<td></td>
<td>• Create/edit/delete tests</td>
</tr>
<tr>
<td></td>
<td>• Create/edit/delete test steps</td>
</tr>
<tr>
<td></td>
<td>• View the step config page</td>
</tr>
<tr>
<td></td>
<td>• View the test runner page</td>
</tr>
<tr>
<td></td>
<td>• View the test suite results, test results and result items pages</td>
</tr>
<tr>
<td></td>
<td>• Execute user tests</td>
</tr>
<tr>
<td></td>
<td>• View, create, edit, delete and execute test suites</td>
</tr>
<tr>
<td></td>
<td>• Create/edit step config records</td>
</tr>
<tr>
<td></td>
<td>• Create/edit Automated Test Framework properties</td>
</tr>
<tr>
<td>atf_test_designer</td>
<td>Has permission to:</td>
</tr>
<tr>
<td></td>
<td>• View the tests page</td>
</tr>
<tr>
<td></td>
<td>• Create/edit/delete tests</td>
</tr>
<tr>
<td></td>
<td>• Create/edit/delete test steps</td>
</tr>
<tr>
<td></td>
<td>• View the step config page</td>
</tr>
<tr>
<td></td>
<td>• View the test runner page</td>
</tr>
<tr>
<td></td>
<td>• View the test suite results, test results and result items pages</td>
</tr>
<tr>
<td></td>
<td>• Execute user tests</td>
</tr>
<tr>
<td></td>
<td>• View, create, edit, delete and execute test suites</td>
</tr>
<tr>
<td></td>
<td>• View Automated Test Framework properties</td>
</tr>
</tbody>
</table>
Tables excluded from rollback after running an automated test

The Automated Test Framework tracks data created by running tests and rolls back changes after testing. The system excludes certain tables from being tracked during testing.

The system excludes certain tables from being tracked or rolled back:

- The *History sets*
- The *ECC Queue table (ecc_queue)*
- The Email *System mailboxes* and *System email log and mailboxes* tables.
- Any table that extends an excluded table

If your test run changes (inserts/updates/deletes) any record on these excluded tables, the system does not roll back the change after testing.

Automated Test Framework Tests module

The Tests module opens the Test table. From here, you can add, edit, and run tests. By opening an individual test record, you can view and edit the steps comprising that test.

Test record form

In the Test record form, you view and edit values of fields for the test record.

<table>
<thead>
<tr>
<th>Field / UI Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Test name</td>
</tr>
<tr>
<td>Description</td>
<td>Test description</td>
</tr>
<tr>
<td>Active</td>
<td>If this test is active <strong>true</strong>. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
<tr>
<td>Test steps related list</td>
<td>The steps that this test executes.</td>
</tr>
<tr>
<td>Test results related list</td>
<td>The results from individual executions of this test.</td>
</tr>
</tbody>
</table>

UI Actions

<table>
<thead>
<tr>
<th>Field / UI Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update</td>
<td>Click to update the test record.</td>
</tr>
<tr>
<td>Run test</td>
<td>Click to run test.</td>
</tr>
<tr>
<td>Copy test</td>
<td>Click to copy test.</td>
</tr>
<tr>
<td>Delete</td>
<td>Click to delete this test.</td>
</tr>
</tbody>
</table>

Automated Test Framework Suites module

The Suites module opens the Test Suites table. You can create, edit, and run test suites from this table.

Test Suite record

The Test suite record contains information about one test suite.
### Fields

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the test suite.</td>
</tr>
<tr>
<td>Active</td>
<td>To make this test suite active, check this field.</td>
</tr>
<tr>
<td>Description (Optional)</td>
<td>Enter a description to identify the purpose of this test suite.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
<tr>
<td>Parent Suite</td>
<td>To make this test suite a child of another test suite, enter the name here.</td>
</tr>
<tr>
<td>Test Suite Tests (related list)</td>
<td>The tests included in this test suite.</td>
</tr>
<tr>
<td>Child Test Suites (related list)</td>
<td>Any test suites that are children of this test suite.</td>
</tr>
<tr>
<td>Test Suite Results (related list)</td>
<td>Results from executing this test suite.</td>
</tr>
<tr>
<td>Test Suite Schedules (related list)</td>
<td>Any test suite schedules that include this test suite.</td>
</tr>
</tbody>
</table>

### Automated Test Framework Test Results module

The Test Results module opens the Test Results table. You can drill down to see details about the results of individual tests and individual steps within a test.

When you run a test, the automated test framework creates a Test Results record that you can inspect to learn what happened.

Test results records appear on a test record as related lists. Each time you run a test, the system adds a new test results record. By default, the system deletes test and test suite results data 30 days after creation. You can modify the default retention policy with the table cleanup module.

Each test results record includes a related list of test results item records, which correspond to the results from individual test steps.

If the test generated screenshots, they appear as attachments to the test results record.

**Test Results record**

The Test results record contains information about one test execution.

### Manage Attachments

If this test created screenshots, they appear as attachments in this section.

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test name</td>
<td>The name of the test.</td>
</tr>
<tr>
<td>Field / Element</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Status               | The result of the test or suite execution:  
  - success - test or suite passed  
  - failure - test or suite failed  
  - waiting - test or suite is waited to start  
  - running - test or suite is running  
  - skipped - occurs if an earlier test in the suite failed and **Abort on failure** is set to **true**.  
  - error - an error occurred in the test framework, for example, the test runner halted or the server encountered an unintended scenario. The error message appears in the summary field of the steps results record for the step that threw the error.  
  - canceled - user canceled the test or suite                                                                                                                                                                                                                                                                                                             |
| Retain indefinitely  | If checked, the system does not delete this record after the specified data retention period has passed. For more information, see **Table Cleanup**.                                                                                                                                                                                                                                                                                                                                                   |
| Start time           | The time this test started executing.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| End time             | The time this test finished executing.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Duration             | The duration it took to execute this test.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Previous test result | Only appears if this test result is for a re-run of a failed test and you are logged on with the admin, atf_test_admin, or atf_test_designer role. Click the information icon to navigate to the "original" test result record.  
  ![Previous test result](image)                                                                                                                                                                                                                                                                                                                                                                                               |
| Output               | Test result (often **passed** or **failed**). For failed tests, includes the summary for the first test step that failed.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Browsers involved    | User agent strings returned by browsers that ran the test.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Step Results         | Contains a record for each step result and log entry in this test.                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Test Log             | Contains a record for each test results item related to this record.                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
## Field / Element
<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Transactions (related list)</td>
</tr>
<tr>
<td>Contains a record for each transaction (from the system transaction log (syslog_transaction) related to this record.</td>
</tr>
<tr>
<td>To view the step results record associated with a transaction, click entry in the step results column.</td>
</tr>
<tr>
<td>To view the transaction log record associated with a transaction, click the value in the transaction record column.</td>
</tr>
</tbody>
</table>

### Note:
The system may not be able to log some transactions with an extremely-short duration.

### Steps Results record
The Step Results record contains information about one step in a test result.

### Fields

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>The result of the test or suite execution:</td>
</tr>
<tr>
<td></td>
<td>- success - test or suite passed</td>
</tr>
<tr>
<td></td>
<td>- failure - test or suite failed</td>
</tr>
<tr>
<td></td>
<td>- waiting - test or suite is waited to start</td>
</tr>
<tr>
<td></td>
<td>- running - test or suite is running</td>
</tr>
<tr>
<td></td>
<td>- skipped - occurs if an earlier test in the suite failed and <strong>Abort on failure</strong> is set to <strong>true</strong>.</td>
</tr>
<tr>
<td></td>
<td>- error - an error occurred in the test framework, for example, the test runner halted or the server encountered an unintended scenario. The error message appears in the summary field of the steps results record for the step that threw the error.</td>
</tr>
<tr>
<td></td>
<td>- canceled - user canceled the test or suite</td>
</tr>
<tr>
<td>Type</td>
<td>The type of test result item: Step Result or Client Log.</td>
</tr>
<tr>
<td>Summary</td>
<td>Contains a summarized version of the output field. If the step threw an error, contains a detailed exception/error message.</td>
</tr>
<tr>
<td></td>
<td>For a Send REST Request - Inbound step, the request endpoint and response code.</td>
</tr>
<tr>
<td>Field / Element</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Output</td>
<td>For a Step Result, the outputs from the step, including any explanation why a step failed or was skipped.</td>
</tr>
<tr>
<td></td>
<td>For a Client Log, the log entry.</td>
</tr>
<tr>
<td></td>
<td>For a Send REST Request - Inbound step, the REST request and response including the response body. The output field is truncated at 4096 characters.</td>
</tr>
<tr>
<td></td>
<td>The REST request and response headers are filtered to prevent sensitive information from being added to the log. A filtered header text is replaced with the text “Header redacted for security.”</td>
</tr>
<tr>
<td></td>
<td>See Filter REST request and response headers for information on how to add a list of REST request and response headers to be filtered.</td>
</tr>
<tr>
<td></td>
<td>For additional console logging and test execution information for a test step, see Test Logs record.</td>
</tr>
<tr>
<td>Step</td>
<td>The name of the step executed. (May be blank for a client log.)</td>
</tr>
<tr>
<td>Description</td>
<td>For a Step Result, the actions performed. For a Client Log, blank.</td>
</tr>
<tr>
<td>Start Time</td>
<td>The time this step or log entry started.</td>
</tr>
<tr>
<td>End Time</td>
<td>The time this step or log entry was completed.</td>
</tr>
<tr>
<td>Duration</td>
<td>The duration between the Start Time and End Time.</td>
</tr>
<tr>
<td>Step Transactions (related list)</td>
<td>Contains a record for each transaction (from the system transaction log (syslog_transaction) related to this record.</td>
</tr>
</tbody>
</table>

**Test Logs record**

The Test Logs record contains console logging and test execution information. During test execution, any information reported to the environment is recorded in the Test Log table. This information can include browser console logging and messages recorded by step environments.

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Time</td>
<td>The time this step or log entry started.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
## Automated Test Framework Suite Results module

The Suites Results module opens the Suites Results table. You can drill down to see details about the results of individual test suites, the individual tests within those test suites, any child test suites, and so on.

**Test Suite Results record**

The Test Suite Results record contains information about the results of one execution of one test suite.

### Fields

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Time</td>
<td>(Appears in list of records, but not on default record form). The time this test suite started.</td>
</tr>
<tr>
<td>Test Suite</td>
<td>The test suite that was run.</td>
</tr>
<tr>
<td>Number</td>
<td>A unique ID for this test suite results record.</td>
</tr>
<tr>
<td>Base Test Suite Result</td>
<td>If this test suite is a child in a hierarchy of test suites, the base test suite result is the unique ID of the result record for the suite at the top-most level of the hierarchy. For more information, see Example: Base test suite result.</td>
</tr>
<tr>
<td>Field / Element</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Parent</td>
<td>If this test suite has a parent, this is the test result record for the parent suite. For more information, see <a href="#">Example: Parent test suite results</a>.</td>
</tr>
<tr>
<td>Status</td>
<td>The result of the test or suite execution:</td>
</tr>
<tr>
<td></td>
<td>· success - test or suite passed</td>
</tr>
<tr>
<td></td>
<td>· failure - test or suite failed</td>
</tr>
<tr>
<td></td>
<td>· waiting - test or suite is waited to start</td>
</tr>
<tr>
<td></td>
<td>· running - test or suite is running</td>
</tr>
<tr>
<td></td>
<td>· skipped - occurs if an earlier test in the suite failed and <a href="#">Abort on failure</a> is set to true.</td>
</tr>
<tr>
<td></td>
<td>· error - an error occurred in the test framework, for example, the test runner halted or the server encountered an unintended scenario. The error message appears in the summary field of the steps results record for the step that threw the error.</td>
</tr>
<tr>
<td></td>
<td>· canceled - user canceled the test or suite</td>
</tr>
<tr>
<td>Run time</td>
<td>The duration it took to execute this test suite.</td>
</tr>
<tr>
<td>Retain indefinitely</td>
<td>If checked, the system does not delete this record after the specified data retention period has passed. For more information, see <a href="#">Table Cleanup</a>.</td>
</tr>
<tr>
<td>Rolled up test success count</td>
<td>How many tests were successful. The tests counted as part of the roll up are all tests included in this suite, plus all others included in suites that are descendents of this one. For more information, see <a href="#">Rolled up counts for test suites results</a>.</td>
</tr>
<tr>
<td>Rolled up test failure count</td>
<td>How many tests failed. The tests counted as part of the roll up are all tests included in this suite, plus all others included in suites that are descendents of this one. For more information, see <a href="#">Rolled up counts for test suites results</a>.</td>
</tr>
<tr>
<td>Rolled up test error count</td>
<td>How many tests resulted in an error. The tests counted as part of the roll up are all tests included in this suite, plus all others included in suites that are descendents of this one. For more information, see <a href="#">Rolled up counts for test suites results</a>.</td>
</tr>
<tr>
<td>Rolled up test skip account</td>
<td>How many tests were skipped. The tests counted as part of the roll up are all tests included in this suite, plus all others included in suites that are descendents of this one. For more information, see <a href="#">Rolled up counts for test suites results</a>.</td>
</tr>
<tr>
<td>Test Results (related list)</td>
<td>Results of the individual tests included in this test suite.</td>
</tr>
<tr>
<td>Child Test Suites Results (related list)</td>
<td>The results of any test suites that are children of this test suite.</td>
</tr>
<tr>
<td>Field / Element</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>All Test Suite Results (related list)</td>
<td>Results from this test suite and all descendent test suites.</td>
</tr>
<tr>
<td>Failed Tests in Suite (related list)</td>
<td>Results from any failed tests included in this test suite.</td>
</tr>
<tr>
<td>Previous suite result</td>
<td>Only appears if this suite result is for a re-run of failed tests and you are logged on with the admin, atf_test_admin, or atf_test_designer role. Click the information icon to navigate to the &quot;original&quot; suite result record.</td>
</tr>
</tbody>
</table>

Test suite results examples
Examples of relationship terms and how aggregated results roll up for test suites.

Example: Base test suite result
The base test suite is the top-level test suite in the hierarchy. For example, if you are viewing the test suite results record for (A) Test Suite 1.3.1, the
**base test suite result** field links to the test suite results record for (B) Test Suite

![Diagram of test suite hierarchy]

1. **Example: Parent test suite results**

The *parent test suite* is the test suite immediately above the one you are currently viewing. For example, if you are viewing the test suite results record for (A) Test Suite 1.3.1, the *parent* field links to the test suite results record for (B) Test Suite 1.3.
Rolled up counts for test suites results

You can view rolled up counts for a test suite for four metrics: test successes, failures, errors, and skips. Each of these sum data from all tests in the test suite plus all tests in the test suite's descendents.

For example, if you are viewing the test suite results record for (A) Test Suite 1.3.1, the Rolled up test success count field shows the total number of successes counting results from all the tests represented by boxes labeled A.

If you are viewing the test suite results record for (B) Test Suite 1.3.1, the Rolled up test success count field shows the total number of successes counting results from all the tests represented by boxes labeled B.
Child Test Suites Results

If you are viewing the test suite results record for (A) Test Suite 1, the Child Test Suites Results related list shows result records for Test Suite 1.3 (B).
All Test Suites Results

If you are viewing the test suite results record for (A) Test Suite 1, the All Suites Results related list shows result records for all Test Suites represented by boxes labeled A.

If you are viewing the test suite results record for (B) Test Suite 1.3, the All Suites Results related list shows result records for all Test Suite 1.3.1 only (represented by the box labeled B).
Automated Test Framework Schedule module

Open the Suites Schedules table. You can drill down to see details about the results of individual schedules or create a new schedule.

_Suite Schedule record_

The Suite Results record contains information about one test suite schedule.

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A name for this test suite schedule.</td>
</tr>
<tr>
<td>Description</td>
<td>A description for this test suite schedule.</td>
</tr>
<tr>
<td>Active</td>
<td>If this schedule is active <strong>true</strong>, Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope for this test suite schedule.</td>
</tr>
<tr>
<td>Run</td>
<td>The frequency with which the system runs test suites belonging to this schedule.</td>
</tr>
<tr>
<td>Run as tz</td>
<td>The timezone used for the <strong>Time</strong> field.</td>
</tr>
<tr>
<td>Field / Element</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Time</td>
<td>The time of day at which the system runs test suites belonging to this schedule.</td>
</tr>
<tr>
<td>Day</td>
<td>If Run is set to Weekly, the day of the week on which the system runs test suites belonging to this schedule. If Run is set to monthly, the day of the month.</td>
</tr>
<tr>
<td>Repeat interval</td>
<td>If Run is set to Periodically, the number of days and hours that constitute the repeat interval for running suites in this schedule.</td>
</tr>
<tr>
<td>Conditional</td>
<td>Check to enable a script to define conditions under which the system should run the test suites in this schedule.</td>
</tr>
<tr>
<td>Condition</td>
<td>If Conditional is checked, the script to execute for determining the conditions under which the system should run the test suites in this schedule.</td>
</tr>
<tr>
<td>Scheduled Suite Runs (related list)</td>
<td>The test suites the system should run on this schedule.</td>
</tr>
</tbody>
</table>

**Scheduled Suite Run record**
A Scheduled Suite Run record associates a Suite Schedule record with a Test Suite.

**Fields**

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
<td>The schedule to use.</td>
</tr>
<tr>
<td>Test Suite</td>
<td>The test suite to run.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope for this scheduled suite run.</td>
</tr>
<tr>
<td>Watch list</td>
<td>Users the system notifies when this scheduled suite run completes.</td>
</tr>
<tr>
<td>Browser name</td>
<td>If the test suite has UI components, the browser that must be used. If no client test runner is available with this browser, the system does not run the suite.</td>
</tr>
<tr>
<td>Browser version starts with</td>
<td>If the test suite has UI components, the browser version for the client test runner must start with this string. If no client test runner is available with this browser version, the system does not run the suite.</td>
</tr>
<tr>
<td>OS name</td>
<td>If the test suite has UI components, the client test runner must run under this OS. If no client test runner is available with this OS, the system does not run the suite.</td>
</tr>
<tr>
<td>OS version starts with</td>
<td>If the test suite has UI components, the OS version for the client test runner must start with this string. If no client test runner is available with this OS version, the system does not run the suite.</td>
</tr>
</tbody>
</table>
To determine the browser name and version of a browser you want to use, start a scheduled test runner with that browser, then inspect that runner’s record in the **Active Scheduled Test Runners Module**.

### Automated Test Framework Run module

Start a client test runner and view information about test runners and test runs.

For information on how to work with test runners, see *Working with client test runners.*

**Automated Test Framework Client Test Runner module**

Open a browser window or tab for running manually-started client-side automated tests.

You can toggle a client test runner to act as either a manual or scheduled client test runner. For more information about scheduling test suites, see *Schedule automated test suite.*

### Fields on the Client Runner Test window

<table>
<thead>
<tr>
<th>Field / UI Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form preferences icon</td>
<td>Click to display the form preferences panel.</td>
</tr>
<tr>
<td>Form preferences panel: Screenshots mode</td>
<td>Choose among: • Enable for all steps • Enable for failed steps • Disable for all steps For additional information, see <em>Set the system property to control when the Automated Test Framework captures screenshots.</em></td>
</tr>
<tr>
<td>Form preferences panel: Run scheduled tests only</td>
<td>Click to toggle between: • On (green): use this client test runner to run only scheduled tests and suites • Off (gray): use this client test runner to run only manually-started tests and suites.</td>
</tr>
</tbody>
</table>
Fields on the Client Runner Test window

<table>
<thead>
<tr>
<th>Field / UI Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form preferences icon</td>
<td>Click to display the form preferences panel.</td>
</tr>
<tr>
<td>Form preferences panel: Screenshots mode</td>
<td>Choose among:</td>
</tr>
<tr>
<td></td>
<td>- Enable for all steps</td>
</tr>
<tr>
<td></td>
<td>- Enable for failed steps</td>
</tr>
<tr>
<td></td>
<td>- Disable for all steps</td>
</tr>
<tr>
<td></td>
<td>For additional information, see  <em>Set the system property to control when the Automated Test Framework captures screenshots.</em></td>
</tr>
<tr>
<td>Form preferences panel: Run scheduled tests only</td>
<td>Click to toggle between:</td>
</tr>
<tr>
<td></td>
<td>- On (green): use this client test runner to run only scheduled tests and suites</td>
</tr>
<tr>
<td></td>
<td>- Off (gray): use this client test runner to run only manually-started tests and suites</td>
</tr>
</tbody>
</table>

*Automated Test Framework Active Manual Test Runners module*

View the client test runners table filtered to show only those runners available to run manually-started tests.

When you start a manual *client test runner*, the system registers that runner in the Test Runners table as *active*.

The data in this table is transient. While the runner is active, it reports in to the system at a specified interval. If the runner does not report in at the expected time, the system marks the runner as *inactive*. After a period of time the system deletes the runner. You can modify these intervals on the *Automated Test Framework properties* page.
### Example Active Manual Client Runner Test table

**Fields on the Client Runner Test table**

<table>
<thead>
<tr>
<th>Field / UI Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Whether this test runner is for manual or scheduled tests. In the Active Manual Test Runners module, Type is always <strong>Manual</strong>.</td>
</tr>
<tr>
<td>User</td>
<td>The user logged into the browser session.</td>
</tr>
<tr>
<td>Browser name</td>
<td>The browser name.</td>
</tr>
<tr>
<td>Browser version</td>
<td>The browser version.</td>
</tr>
<tr>
<td>OS name</td>
<td>The name of the operating system running the browser.</td>
</tr>
<tr>
<td>OS version</td>
<td>The version of the operating system running the browser.</td>
</tr>
</tbody>
</table>
### Field / UI Element

<table>
<thead>
<tr>
<th>Field / UI Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Whether this runner is currently online or offline. In the Active Manual Test Runners module, Status is always Online.</td>
</tr>
<tr>
<td>Last checkin</td>
<td>The time/date this runner most recently reported in to the system.</td>
</tr>
<tr>
<td>Status reason</td>
<td>If the Status is Offline, the reason why. In the Active Manual Test Runners module, Status reason is always empty.</td>
</tr>
</tbody>
</table>

**Automated Test Framework Active Scheduled Test Runners module**

View the client test runners table filtered to show only those runners available to run tests to be started by a schedule.

When you start a **scheduled client test runner**, the system registers that runner in the Active Scheduled Test Runners table.

The Active Scheduled Test Runner module is useful when you create a scheduled suite run. For scheduled suite runs, you can specify the browser to use. To determine the name and version of a browser you want to use, start a scheduled test runner with that browser, then inspect that runner’s record in the Active Scheduled Test Runners module.

The data in this table is transient. While the runner is active, it reports in to the system at a specified interval. If the runner does not report in at the expected time, the system marks the runner as inactive. After a period of time the system deletes the runner.

You can **modify these intervals** on the **Automated Test Framework properties** page.

---

**Example Scheduled Client Runner Test table**
Fields on the Client Runner Test table

<table>
<thead>
<tr>
<th>Field / UI Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Whether this test runner is for manual or scheduled tests. In the Active Scheduled Test Runners module, Type is always <strong>Scheduled</strong>.</td>
</tr>
<tr>
<td>User</td>
<td>The user logged into the browser session.</td>
</tr>
<tr>
<td>Browser name</td>
<td>The browser name.</td>
</tr>
<tr>
<td>Browser version</td>
<td>The browser version.</td>
</tr>
<tr>
<td>OS name</td>
<td>The name of the operating system running the browser.</td>
</tr>
<tr>
<td>OS version</td>
<td>The version of the operating system running the browser.</td>
</tr>
<tr>
<td>Status</td>
<td>Whether this runner is currently online or offline. In the Active Manual Test Runners module, Status is always Online.</td>
</tr>
<tr>
<td>Last checkin</td>
<td>The time/date this runner most recently reported in to the system.</td>
</tr>
<tr>
<td>Status reason</td>
<td>If the Status is Offline, the reason why. In the Active Scheduled Test Runners module, Status reason is always empty.</td>
</tr>
</tbody>
</table>

**Automated Test Framework Waiting/Running Test Runs module**
The Waiting/Running Test Run module opens a list of records showing the tests waiting to be run.

**Note:** To prevent conflicts, the system allows only one test to run at a given time. This is true even if you have multiple client test runner windows open. If you submit tests to run when another test is already running, the system holds the new tests to run later. If a test remains waiting for more than ten minutes, the system cancels the test.

You can **cancel execution of a waiting automated test**.

**Automated Test Framework Waiting/Running Suite Runs module**
The Waiting/Running Suite Runs module opens a list of records showing the test suites waiting to be run.

**Automated Test Framework Administration modules**
The administration module contains forms for configuring and managing the automated test framework.

**Automated Test Framework Properties**
On the Properties form, you can set parameters that control how the system executes automated tests and test suites.

**Assigned ATF Role**

<table>
<thead>
<tr>
<th>Assigned ATF Role</th>
<th>Role Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>atf_test_admin</td>
<td>Create or edit Automated Test Framework properties</td>
</tr>
</tbody>
</table>
### Assigned ATF Role

<table>
<thead>
<tr>
<th>Role Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>View Automated Test Framework properties only (cannot create or edit properties)</td>
</tr>
</tbody>
</table>

### Note:
For detailed information about ATF roles and their associated Create, Edit or View permissions, refer to *Automated Test Framework user roles*.

### Test/Test Suite Properties

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable test/test suite execution.</td>
<td>If checked, enables running tests and test suites on this instance. This setting is unchecked by default to prevent users from unintentionally running tests on production instances.</td>
</tr>
<tr>
<td>Enable scheduled test suite execution</td>
<td>If checked, enables scheduling test suites on this instance.</td>
</tr>
</tbody>
</table>

### Test Debugging Properties

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enables additional debugging functionality . . .</td>
<td>Enables additional debugging functionality, including adding a debugging tab on the client Test Runner UI page and saving UI Test Result JSON to the test result record.</td>
</tr>
</tbody>
</table>

### Screenshots Capture Mode.

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
</table>
| Enable or disable screenshot capture during test execution. | - To capture screenshots for all steps, select *Enable for all steps*.  
  - To capture screenshots only for failed steps, select *Enable for all failed steps*.  
  - To capture no screenshots, select *Disable for all steps*. |

### Test Runner Properties

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test runner heartbeat interval</td>
<td>Time interval in seconds for sending a heartbeat from the test runner to the server.</td>
</tr>
<tr>
<td>Offline test runner retention interval</td>
<td>If an offline test runner does not communicate with the system for this period of days, the system deletes that test runner.</td>
</tr>
</tbody>
</table>
### Test Suite Report Properties

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test suite report properties</td>
<td>The number of test suite results to display in the test suite aging report.</td>
</tr>
</tbody>
</table>

### Email Properties

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean value for results displayed in scheduled suite result emails.</td>
<td>When true, the scheduled suite test result emails only show results that failed. When false, displays all results.</td>
</tr>
<tr>
<td>Maximum number of test results to be displayed in scheduled suite result emails.</td>
<td>Maximum number of test results to be displayed in scheduled suite result emails.</td>
</tr>
<tr>
<td>Maximum depth when printing suite results in suite result emails.</td>
<td>Maximum depth when printing suite results in suite result emails.</td>
</tr>
<tr>
<td>Color to indicate an ATF test failed in scheduled suite result emails.</td>
<td>Hexadecimal code for color to indicate an ATF test failed in scheduled suite result emails.</td>
</tr>
<tr>
<td>Color to indicate an ATF test errored in scheduled suite result emails.</td>
<td>Hexadecimal code for color to indicate an ATF test errored in scheduled suite result emails.</td>
</tr>
<tr>
<td>Color to indicate an ATF test passed in scheduled suite result emails.</td>
<td>Hexadecimal code for color to indicate an ATF test passed in scheduled suite result emails.</td>
</tr>
<tr>
<td>Color to indicate an ATF test was skipped in scheduled suite result emails.</td>
<td>Hexadecimal code for color to indicate an ATF test was skipped in scheduled suite result emails.</td>
</tr>
<tr>
<td>Color to indicate an ATF test was canceled in scheduled suite result emails.</td>
<td>Hexadecimal code for color to indicate an ATF test was canceled in scheduled suite result emails.</td>
</tr>
</tbody>
</table>

### Automated Test Framework Step Configurations module

Step configurations are the records that define how each type of step behaves. Automated Test Framework Step Config record

The step config record controls how a test step of this type behaves.

### Fields

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name for this test step configuration</td>
</tr>
<tr>
<td>Active</td>
<td><strong>True</strong> if this step configuration is active. Otherwise, <strong>false</strong>.</td>
</tr>
<tr>
<td>Step Environment</td>
<td>The <em>step environment</em> in which a step with this configuration can run.</td>
</tr>
<tr>
<td>Category</td>
<td>The <em>category</em> assigned to a step with this configuration.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs steps with this configuration.</td>
</tr>
<tr>
<td>Field / Element</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Batch order constraint</td>
<td>Constrains where a step with this configuration can appear in a test:</td>
</tr>
<tr>
<td></td>
<td>· <strong>None</strong>: a step based on this configuration can appear at any point in a test</td>
</tr>
<tr>
<td></td>
<td>· <strong>Start Batch Execution</strong>: if a test includes this a step based on this configuration, it must be the first step in the test</td>
</tr>
<tr>
<td></td>
<td>· <strong>Run in the middle of an execution</strong>: if a test includes a step based on this configuration, it can't be the first or last step.</td>
</tr>
<tr>
<td></td>
<td>· <strong>Stop Execution</strong>: if a test includes a step based on this configuration, it must be the last step in the test.</td>
</tr>
<tr>
<td>Class type</td>
<td>For custom step configurations, this field is always <strong>Script</strong>.</td>
</tr>
<tr>
<td>Order</td>
<td>An integer specifying where steps with this configuration appear in the step list on the Add Test Step dialog. For more information, see the <strong>example using the Order field</strong>.</td>
</tr>
<tr>
<td>Template reminder</td>
<td>The instructions that appear when a step with this configuration is included in a test as part of a template. For more information, see the <strong>example of using the Template reminder field</strong>.</td>
</tr>
<tr>
<td>HTML description</td>
<td>The text that appears when the cursor highlights a step with this configuration on the Create New Step dialog. For more information, see the <strong>example using the HTML description field</strong>.</td>
</tr>
<tr>
<td>Description generation script</td>
<td>Generates the text that describes a step when a test includes it. For more information, see the <strong>example using the description generation script</strong>.</td>
</tr>
<tr>
<td>Step execution script</td>
<td>Script (javascript) that runs when a step with this config runs.</td>
</tr>
<tr>
<td>Input Variables related list</td>
<td>The variables that act as inputs for a step with this config.</td>
</tr>
<tr>
<td>Output variables related list</td>
<td>The variables that act as outputs for a step with this config.</td>
</tr>
</tbody>
</table>

**Examples of step config field values**

Examples of where the system displays values assigned to some of the step config fields.

**Order field example**

The steps in the middle column are sorted according to the values of the **Order** field.
Category field example

The labels in the left column show the categories for available steps. You can filter the list in the middle column, by selecting a category in the left column.

Template reminder example

Here is a portion of the new record form for a test step config for Example Custom Step.
When you add a template to a test, the system generates a set of instructions for completing the template steps and saves them in the test **Description** field. The text in the step config’s **Template reminder field** appears as the instructions for the corresponding step.
HTML description example

Here is a portion of the new record form for a custom step named *Example Custom Step*.
Here is how the step appears in the Add Test Step dialog.
Description generation script example

Here is a portion of the new record form for a custom step named *Example Custom Step*. 
When a step of this type is included in a test, the generated description appears in the Test Steps related list.
Step execution scripts

In a step configuration record, the step execution script field determines what a step with this configuration does when it runs.

Step inputs

The input variables to a step are determined by the inputs related list in the step configuration record. The inputs parameter to executeScript() gives the script access to these variables. For example, if the inputs related list contains two records, var1 and var2, the script can reference var1 with the expression inputs.var1 and can reference var2 with inputs.var2.

Step outputs

The output variables to a step are determined by the outputs related list in the step configuration record. The outputs parameter to executeScript() gives the script access to these variables. For example, if the outputs related list contains two records, out1 and out2, the script can reference out1 with the expression outputs.out1 and can reference out2 with outputs.out2.
**Step result**

The `stepResult` parameter provides access to an API that controls whether the step passes or fails. It also determines the message the step writes to the log.

The method `stepResult.setSuccess()` causes the step to succeed. The method `stepResult.setFailed()` causes the step to fail.

The method `stepResult.setOutputMessage()` sets the message to write to the log when the step succeeds or fails. It takes one parameter: the string to write to the log. If the script calls `stepResult.setOutputMessage()` more than once, the most recent value set overwrites any previous value.

**Record Query step execution script**

```javascript
(function executeStep(inputs, outputs, stepResult) {
  if (gs.nil(inputs.table)) {
    stepResult.setOutputMessage(gs.getMessage("The '{0}' input variable was not specified", 'table"));
    stepResult.setFailed();
    return;
  }
  var query = new GlideRecord(inputs.table);
  query.addEncodedQuery(inputs.field_values);
  query.query();
  if (!query.next()) {
    stepResult.setOutputMessage(gs.getMessage("No records matching query:
    {0}", inputs.field_values));
    stepResult.setFailed();
  } else {
    stepResult.setSuccess();
    outputs.table = inputs.table;
    outputs.first_record = query.getUniqueValue();
    stepResult.setOutputMessage(gs.getMessage("Found {0} [1] records matching query:
    \n    {2}"),
    [query.getRowCount(), inputs.table, inputs.field_values]));
  }
})(inputs, outputs, stepResult);
```

**Step description generation script**

In a step configuration record, the step description generation script field determines the step description that the system generates when a step of this type is added to a test.

For an example showing how where the description generated by this script appears, see [Description generation script example](#).

---

© 2020 ServiceNow, Inc. All rights reserved.

ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Step

The `step` parameter to `generateDescription()` gives the script access to the step object, which it turn gives access to the input variables as defined in the step configuration record. (Input variables are defined in a related list.)

For example, if in a step config record – the inputs related list contains two records: `var1` and `var2`, the script can reference `var1` with the expression `step.inputs.var1` and can reference `var2` with `step.inputs.var2`.

**Record Query description generation script**

```javascript
(function generateDescription(step) {
  var td = GlideTableDescriptor.get(step.inputs.table);
  if (!td) {
    gs.log("Invalid table name in test step: " +
     step.inputs.table);
    return gs.getMessage("Set field values");
  }
  var descriptionGenerator = new
  ATFStepDescriptionGenerator();
  var description = gs.getMessage("There should be at least
  one record in '{0}' matching " +
    "a query of\n{1}",
    [step.inputs.table.getDisplayValue(),
     descriptionGenerator.getConditionDescription(step.inputs.table,
     step.inputs.field_values)]);

  return description;
})(step);
```

**Automated Test Framework Step Environments module**

A test step environment specifies where the step executes (for example, server versus browser). In this release, custom step configs can use only the Server-Independent environment.

In this release, you cannot add custom test step environments.

**Automated Test Framework Table Cleanup**

The Table Cleanup module opens a list of records specifying the retention policies for test result and test suite result tables and the records within them. From this module, you can view and modify these policies.

By default, the system deletes test and test suite results data 30 days after creation. To modify the retention policies for a table and its records, click the table for which you want to modify policies. For more information, see Modify data retention policy for ATF test results.

**Automated Test Framework autoflush form**

On the Auto Flush form, you specify a retention policy for a set of records on a given test results table.

**Fields**

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tablename</td>
<td>The table containing the records to which the policy applies.</td>
</tr>
<tr>
<td>Matchfield</td>
<td>The field for which the system monitors duration.</td>
</tr>
<tr>
<td>Field / Element</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Age in seconds</td>
<td>The amount of time (in seconds) the system waits before deleting the records.</td>
</tr>
<tr>
<td>Active</td>
<td>True if this policy is active. Otherwise, false.</td>
</tr>
<tr>
<td>Application scope</td>
<td>The application scope to which this policy applies.</td>
</tr>
<tr>
<td>Cascade delete</td>
<td>If checked, the system deletes all matching records plus any records referring to them. If unchecked, the system deletes matching records but not records referring to them.</td>
</tr>
<tr>
<td>Conditions</td>
<td>The filter conditions defining the records in this table to which the policy applies.</td>
</tr>
</tbody>
</table>

**Automated Test Framework Test Templates module**

The Test Templates module opens a list of available templates. From this module, you create, view, and edit test templates.

To view or edit a test template, click the row for the template you want. For more information, see *Edit automated test steps template*.

For information on how to add a new template, see *Create automated test steps template*.

For information on how to use a template when creating a test, see *Add a predefined list of steps (template) to an automated test*.

**Automated Test Template record**

The Test Template record contains information about one test template.

**Fields**

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of this test template.</td>
</tr>
<tr>
<td>Application</td>
<td>The application scope in which the system runs this test or test suite.</td>
</tr>
<tr>
<td>Test Template</td>
<td>The tests to include in this template.</td>
</tr>
<tr>
<td>Description</td>
<td>(Optional) Enter a description to identify the purpose of this test template.</td>
</tr>
</tbody>
</table>

**Automated Test Framework Step Configuration Categories module**

The Step Configuration Categories module opens a list of records specifying the step categories on the Add Step dialog. From this module, you can add, delete, and edit these categories.

Categories are used for filtering the step list in the Add Step dialog. For more information, see *Category field example*.

**Automated Test Framework Test Step Config Category form**

On the Test Step Config Category form, you specify a retention policy for a set of records on a given test results table.
### Fields

<table>
<thead>
<tr>
<th>Field / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name for this step config category.</td>
</tr>
<tr>
<td>Step Environment</td>
<td>The step environment for this step config category.</td>
</tr>
<tr>
<td>Display name</td>
<td>The category name that appears in the middle column of the Add Test Step dialog when this category is selected. See example of <em>Step Config Category Display name</em>.</td>
</tr>
<tr>
<td>Application scope</td>
<td>The application scope to which this policy applies.</td>
</tr>
</tbody>
</table>

### ServiceNow Studio

ServiceNow Studio provides an IDE-like interface for application developers to work on custom applications in one centralized location. It offers a simple way to identify and interact with application files, create files as you develop, and modify existing application files in a tabbed environment.

The system opens Studio whenever you edit a scoped application.
With Studio, application developers can:

- See exactly what files comprise their application in the **Application Explorer**.
- Add new files to their application using a single **Create Application File** interface.
- Navigate to files using familiar search-by-name or by-type behavior with the **Go To** dialog.
- Find code both within and outside an application using the **Code Search** tool.
- Operate on multiple files at once using the tabbed interface.
- Operate on multiple applications at once using multiple studio windows.
- Publish the application to company instances or the ServiceNow Store.
- View information about the current application from the **Status Bar**.

**Note:** Studio is not intended for global applications and can behave unexpectedly when editing them.

### Parts of the Studio UI

<table>
<thead>
<tr>
<th>UI element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>Displays menus and controls.</td>
</tr>
<tr>
<td><strong>File Menu</strong></td>
<td>Contains a list of application-specific options.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Create File</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Import From Source Control</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Create Application</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Publish</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Settings</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Switch</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Manage Developers</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Launch Script Debugger</strong></td>
</tr>
<tr>
<td><strong>Source Control Menu</strong></td>
<td>Contains a list of source control options.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Link To Source Control</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Edit Repository Configuration</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Apply Remote Changes</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Commit Changes</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Stash Local Changes</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Switch Branch</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Create Branch</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Create Tag</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Manage Stashes</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>View History</strong></td>
</tr>
<tr>
<td><strong>Window Menu</strong></td>
<td>Contains a list of tab management options.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Close Current Tab</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Close All Tabs</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Close Other Tabs</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Close Unmodified Tabs</strong></td>
</tr>
<tr>
<td><strong>Search Menu</strong></td>
<td>Contains a list of search options.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Go To</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Code Search</strong></td>
</tr>
<tr>
<td>User name</td>
<td>The header displays the name of current user.</td>
</tr>
<tr>
<td><strong>Create Application File</strong></td>
<td>Allows developers to add an application file to an application.</td>
</tr>
<tr>
<td>Go To</td>
<td>Search for application files by name or type.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>UI element</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Code Search**     | Search within application files for a text string. Search options include:  
|                     |   • Restrict search to a particular table  
|                     |   • Include all applications  
|                     |                                                                                                                                              |
| **Application Explorer** | Displays a list of application files by type. Resize the Application Explorer to see more about application files or to provide more space for the content frame. |
| **Collapse All**    | Collapses all nodes in the application explorer.                                                                                               |
| **Expand All**      | Expand all nodes in the application explorer.                                                                                                  |
| **Data Model > Tables** | A list of application tables. Click a table name to display and edit it in the content frame.                                               |
| **Access Control**  | A list of application access elements such as:  
|                     |   • Roles  
|                     |   • Access Controls  
|                     | Click a record name to display and edit it in the content frame.                                                                               |
| **Navigation**      | A list of application navigational elements such as:  
|                     |   • Application Menus  
|                     |   • Modules  
|                     |   • Application Menus (Mobile)  
|                     |   • Modules (Mobile)  
|                     | Click a record name to display and edit it in the content frame.                                                                               |
| **Content frame**   | Displays a detail form for each record in its own tabs.                                                                                       |
| **Welcome to Studio** | A list of keyboard shortcuts.                                                                                                                  |
| **Tabs**            | Each tab contains a specific application file record identified by the record name and file type.  
|                     | Click a tab to display and edit the record.                                                                                                   
|                     | A tab with a blue circle icon indicates that the record contains unsaved changes.                                                             |
| **Status Bar**      | Displays information about the application and the source control integration.                                                               |
| **Application name** | The status bar displays the name of the current application.                                                                                   |
| **Application version** | The status bar displays the current application version.                                                                                      |
| **Total files**     | The status bar displays the total number of application files.                                                                                  |
## Access Studio

Application developers access Studio to create, import, or open applications.

**Role required:** admin or a delegated developer role

1. Select the path available to your role.

### Paths to Studio

<table>
<thead>
<tr>
<th>User role</th>
<th>Action to open Studio</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>Navigate to System Applications &gt; Studio.</td>
</tr>
<tr>
<td>delegated_developer</td>
<td>Navigate by URL to <a href="https://instance.service-now.com/$studio.do">https://instance.service-now.com/$studio.do</a></td>
</tr>
</tbody>
</table>

The system opens Studio in a new browser tab and displays the Load Application page.
2. To create an application, click **Create Application**. The system displays a list of application creation options.

3. To import an existing application from source control, click **Import From Source Control** list. Studio opens Import From Source Control page.

4. To view an existing application, click the application name from the **Applications** list. Studio opens the selected application.

**ServiceNow Studio keyboard shortcuts**

ServiceNow Studio supports various keyboard shortcuts to manage and edit application files.

### Studio keyboard shortcuts

<table>
<thead>
<tr>
<th>Keyboard combination</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Windows**: Control+Shift+O  
**Mac**: Command+Shift+O | **Go To**: Open any file in your application. |
| **Windows**: Control+Shift+C  
**Mac**: Command+Shift+C | **Create New**: Create an application file. |
<table>
<thead>
<tr>
<th>Keyboard combination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows: Control+Shift+F</td>
<td>Code Search. Search for any file or value.</td>
</tr>
<tr>
<td>Mac: Command+Shift+F</td>
<td></td>
</tr>
<tr>
<td>Windows: Control+Shift+X</td>
<td>Close Tab. Close the current tab. If the tab contains unsaved changes, the system prompts the user to save them.</td>
</tr>
<tr>
<td>Mac: Command+Shift+X</td>
<td></td>
</tr>
</tbody>
</table>

### Add an application file to an application

Studio allows application developers to add new application files by type.

**Role required:** admin

This procedure requires creating a scoped application.

You can add application files to update the features of a custom application.

1. Navigate to **System Applications > Applications**.
2. From the **Develop** tab, click the **Edit** button next to the application you want to modify. The system opens the application in the Studio.
3. From the content frame, click **Create Application File**. You can also use a Studio keyboard shortcut.

### Studio keyboard shortcut

<table>
<thead>
<tr>
<th>Keyboard combination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac: Command+Shift+C</td>
<td></td>
</tr>
</tbody>
</table>

Studio opens the Create New Application File pop-up window.
4. Find the application file type you want to create.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search by application file name</td>
<td>In the Filter entry field, enter the name of the application file.</td>
</tr>
<tr>
<td>Search by category</td>
<td>From the left pane, select a category name.</td>
</tr>
</tbody>
</table>

5. From the results pane, select the application file type you want to create.
6. Click Create.
The system displays a blank form for the application file type in a new studio tab.

Enter the necessary fields for the particular application file type you selected.

**Search for an application file by name or type**

Application developers can use Studio to search for application files.

Role required: admin

This procedure requires creating a scoped application.

You can search for application files to add, remove, or update the features of a custom application.

1. Navigate to **System Applications > Applications**.
2. From the Develop tab, click the Edit button next to the application you want to modify.
   The system opens the application in the Studio.
3. From the header, click Go To.
   You can also use a Studio keyboard shortcut.

<table>
<thead>
<tr>
<th>Studio keyboard shortcut</th>
<th>Keyboard combination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Windows</strong>: Control+Shift+O</td>
<td>Go To. Open any file in your application.</td>
</tr>
<tr>
<td></td>
<td><strong>Mac</strong>: Command+Shift+O</td>
<td>Go To. Open any file in your application.</td>
</tr>
</tbody>
</table>

Studio opens the Go To window.

4. Enter a search string.

![Go To window](image)

Studio displays a list of matching application files as you type.

5. From the list of search results, click a record name.
   Studio opens the application file record in a new tab in the content frame.

**Search within application files**

Studio allows application developers to search within application files for matching record values.

Role required: admin

This procedure requires creating a scoped application.
You can search within application files to add, remove, or update application file values.

1. Navigate to System Applications > Applications.
2. From the Develop tab, click the Edit button next to the application you want to modify. The system opens the application in the Studio.
3. From the header, click Code Search.
   You can also use a Studio keyboard shortcut.

   **Studio keyboard shortcut**

<table>
<thead>
<tr>
<th>Keyboard combination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows: Control+Shift+F</td>
<td>Code Search. Search for any file or value.</td>
</tr>
<tr>
<td>Mac: Command+Shift+F</td>
<td>Code Search. Search for any file or value.</td>
</tr>
</tbody>
</table>

   Studio opens the Search pop-up window.

4. In Search term, enter a search string.

5. Optional: Select any additional search criteria.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a table to search</td>
<td>Search for matches only within the selected file type.</td>
</tr>
<tr>
<td>Search in all applications</td>
<td>Search for matches throughout the instance, not just within the current application.</td>
</tr>
</tbody>
</table>

   **Note:** Searches across all applications can take a long time.

6. Click Search.
   The Studio conducts a case-insensitive search of the application files you selected. While the search is running, Studio displays a search progress indicator. You can click the cancel icon to stop the search. When the search is complete, the system opens a new tab in the content frame to display the search results by application file type. Each application file type displays the number of matching search results.
7. From the search results tab, expand an application file type and click a record name. Studio opens the application file record in a new tab in the content frame.

**Update a custom application record**

You can update a custom application record to add new features or change application functionality.

Role required: admin or a delegated developer role granting full access

You can only update applications in development on your local instance. You cannot edit applications downloaded from your company application repository or the ServiceNow Store.

1. Navigate to **System Applications > Applications > Develop**.
2. Click the application name or the **Edit** button for the application you want to update. The system displays the application and application files in Studio.
3. Click **File > Settings**. Studio opens a tab containing the Custom Application record for the current application.
4. Fill in the fields, as appropriate.

**Custom Application form**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>(Required) Enter a label for the application. Changing the name does not change any other field value derived from the application name such as the <strong>Scope</strong>, <strong>Menu</strong>, or <strong>User role</strong> fields.</td>
</tr>
<tr>
<td>Version</td>
<td>Enter version information for the application. Both the ServiceNow application repository and the store use this value to determine whether updates are available to your application users.</td>
</tr>
</tbody>
</table>

**Note**: To publish the application in the ServiceNow Store, the version must conform to the application certification standards.

**Application Scoping**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>(Read Only) Displays the unique application scope set during the creation process. You can change this value only by deleting and recreating the application with a new value. For more information about the protections offered, see <strong>Application scope</strong>.</td>
</tr>
</tbody>
</table>

**Application administration**

Select whether to protect sensitive application data by restricting how users acquire application-specific roles. See **Application administration**.

**Design and Runtime**
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Mode</td>
<td>Select the JavaScript standard the application supports. Select <strong>ES5 Standards Mode</strong> to support features in ECMAScript 5th edition. Select <strong>Compatibility Mode</strong> to support earlier ECMAScript editions.</td>
</tr>
<tr>
<td>Runtime Access Tracking</td>
<td>Select how the application handles script access requests to resources in other applications. Select <strong>None</strong> to authorize all access requests to cross-scope resources without logging them. Select <strong>Tracking</strong> to log and authorize all access requests to cross-scope resources. Select <strong>Enforcing</strong> to log access requests to cross-scope resources but require an administrator to authorize each request.</td>
</tr>
<tr>
<td>Restrict Table Choices</td>
<td>Clear the option to allow the application to see tables from other application scopes. Select the option to restrict design choices to only tables in the same application.</td>
</tr>
<tr>
<td>Subscription Management</td>
<td></td>
</tr>
<tr>
<td>Licensable</td>
<td>Specifies whether the application is tracked by the Subscription Management application.</td>
</tr>
<tr>
<td>Subscription requirement</td>
<td>Not applicable for ServiceNow customers who build custom applications for their own use. Used only by partners who sell and monitor the usage of resellable applications on the ServiceNow Store. Specifies whether the application requires a separate subscription (<strong>Required</strong>) or is monitored only.</td>
</tr>
<tr>
<td>Subscription model</td>
<td>Not applicable for ServiceNow customers who build custom applications for their own use. Used only by partners who sell and monitor the usage of resellable applications on the ServiceNow Store. Specifies how the Subscription Management application tracks usage. See <strong>Types of subscriptions</strong>.</td>
</tr>
<tr>
<td>Primary Menu</td>
<td></td>
</tr>
<tr>
<td>Menu</td>
<td>Select the application menu where you want to display modules. For more information about menus and modules, see <strong>Create an application menu</strong>.</td>
</tr>
<tr>
<td>End user access</td>
<td></td>
</tr>
<tr>
<td>User role</td>
<td>Select the user role required to access the application menu. For more information about user roles, see <strong>Create a role</strong>.</td>
</tr>
<tr>
<td>Short description</td>
<td>Enter a description of the application purpose or usage.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Logo</td>
<td>Select the image the system displays in the applications list and ServiceNow Store.</td>
</tr>
<tr>
<td>Application Files</td>
<td>View configuration records associated with this application in platform feature tables.</td>
</tr>
<tr>
<td>Dependencies</td>
<td>View or add tables or applications on which this application depends. The system automatically adds records to this list when you extend tables or when another application creates application files for this application. Add script-based dependencies. See <a href="#">Dependencies for custom applications</a>.</td>
</tr>
<tr>
<td>Cross scope privileges</td>
<td>View or create cross-scope privilege records to determine which script operations and targets the system allows to run. See <a href="#">Cross-scope privilege record</a>.</td>
</tr>
<tr>
<td>Design Access</td>
<td>View or specify which other applications have design access to tables or records in this application. See <a href="#">Application design access record</a>.</td>
</tr>
</tbody>
</table>

5. Click **Update**.

### Switch between applications

Application developers can switch between applications without leaving the Studio environment. Before starting this procedure, you must create at least one custom application with its own application scope.

Role required: admin

The contextual development environment restricts some changes when the application files belong to another application. Switching to the application that owns the application files ensures that you change the proper application.

1. Navigate to **App Options Menu > Switch Applications**. The system displays the list of applications.
2. Click the application you want to switch to. The system reloads Studio to display the selected application.

### Automatic recovery of draft records

Studio can maintain a version of any open existing record with unsaved changes. Users can recover unsaved changes when their user session ends unexpectedly due to network latency, session timeout, or service interruption.

Automatic recovery only applies to:

- Records open in Studio. The system does not save all draft records.
- Changes made in desktop (non-mobile) browsers. The system does not save draft records from mobile browsers.
- Changes made to existing records. The system does not save draft changes to new records.
• Records containing unsaved changes that are the most recent update to the record. The system discards draft changes when another user has updated the same record.
• Records for tables that extend the Application File (sys_metadata) table.

After the user re-establishes a session, Studio displays a message for each record with recovered changes.

Automatic recovery message

For each recovered record, users can:
• Continue editing and save the record.
• Clear the changes from the recovery cache.

The system automatically clears changes from the recovery cache when a user:
• Saves the record. The system removes the saved record from the recovery cache.
• Confirms navigating away from a record without saving changes. The system removes the abandoned record from the recovery cache.
• Reaches the recovery cache limit of 5 MB of changes. The system removes the record with the oldest update date-time stamp.

By default, automatic recovery is enabled for all Application File (sys_metadata) tables while working from Studio.

Administrators can configure automatic recovery properties to:
• Disable or re-enable automatic recovery.
• Specify a list of field types to exclude from automatic recovery.

Users can enable or disable automatic recovery as a user preference.

Auto recovery properties

Administrators can configure how Studio handles the recovery of draft records by navigating to Auto Recovery > Properties.

Auto recovery properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>glide.ui.auto.recovery</td>
<td>By default, automatic recovery is enabled for all Application File (sys_metadata) tables while working from Studio. Set this property to disable automatic recovery of records.</td>
</tr>
</tbody>
</table>
Auto recovery dictionary attribute

Administrators can configure how the Studio handles the recovery of draft records with a dictionary attribute.

<table>
<thead>
<tr>
<th>Dictionary attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exclude Auto Recovery</td>
<td>Disables automatic recovery of draft records for this table.</td>
</tr>
<tr>
<td>exclude_auto_recovery</td>
<td></td>
</tr>
</tbody>
</table>

Source control integration

The source control integration allows application developers to integrate with a GIT source control repository to save and manage multiple versions of an application from a non-production instance.

Linking an application to source control allows all application developers on a non-production instance to:

- Import applications from a GIT repository
- Pull and apply remote changes from a GIT repository
- Commit all local changes on the instance to a GIT repository
- Create tags to permanently link to a given version of an application
- Create branches to maintain multiple versions of an application simultaneously

To link an application on a non-production instance to source control:

- The non-production instance must have network access to one or more GIT repositories.
- Each application must have its own dedicated GIT repository.
- The repository user credentials must grant developers read and write access to the repository.

**Note:** All application developers on the instance share a single set of credentials per repository.

After linking an application to source control, application developers can use Studio to manage the repository. From Studio, developers can:

- Edit the application repository credentials
- Commit all local changes on the instance
- Apply remote changes from the repository
- Create a branch
- Switch branches
- Import an application from a remote repository

Source control integration does not support managing applications on a production instance. Instead, manage applications on a production instance using the application repository, an update set, or the ServiceNow Store.

ServiceNow does not support modifications to the files in an application git repository outside of an instance. To prevent this, the system generates and adds a checksum.txt file to the git repository. If a file is modified outside of the instance, an error message similar to the following appears when attempting to import from source control:

checksum.txt is missing or contains an invalid checksum. Ensure your repository contains a valid, unmodified source image exported by an instance.

If you receive this error, revert to an earlier commit by following the instructions in the README.md file.

Available source control operations

The source control integration primarily supports operations from Studio, but can also support some operations directly from the GIT repository.

Available source control operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Available from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import from Source Control</td>
<td>Imports an application from the repository to the local instance.</td>
<td>Studio</td>
</tr>
<tr>
<td>Link to Source Control</td>
<td>Allows developers to manage application changes from a GIT repository.</td>
<td>Studio</td>
</tr>
<tr>
<td>Edit Repository Configuration</td>
<td>Updates the GIT repository user credentials.</td>
<td>Studio</td>
</tr>
<tr>
<td>Apply Remote Changes</td>
<td>Updates the local version of the application to match the repository version.</td>
<td>Studio</td>
</tr>
<tr>
<td>Commit Changes</td>
<td>Updates the repository version of the application to match the local version.</td>
<td>Studio</td>
</tr>
<tr>
<td>Stash Local Changes</td>
<td>Removes and saves local changes for later work.</td>
<td>Studio</td>
</tr>
<tr>
<td>Switch Branch</td>
<td>Updates the local version of the application to match the repository branch version.</td>
<td>Studio</td>
</tr>
<tr>
<td>Create Branch</td>
<td>Creates a branch in the repository to save a different version of the application.</td>
<td>Studio, GIT repository</td>
</tr>
<tr>
<td>Create Tag</td>
<td>Creates a tag in the repository to link to a particular application version.</td>
<td>Studio, GIT repository</td>
</tr>
<tr>
<td>Operation</td>
<td>Description</td>
<td>Available from</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Manage Stashes</td>
<td>Allows developers to apply or delete stashed changes.</td>
<td>Studio</td>
</tr>
<tr>
<td>Create repository</td>
<td>Creates a repository to store application changes</td>
<td>GIT repository</td>
</tr>
<tr>
<td>Create credentials</td>
<td>Creates credentials to the repository.</td>
<td>GIT repository</td>
</tr>
<tr>
<td>Grant access to repository</td>
<td>Provides read and write access to the repository tied to a specific set of credentials.</td>
<td>GIT repository</td>
</tr>
</tbody>
</table>

**Link an application to source control**

Linking an application to source control allows application developers to manage changes from a GIT repository.

- **Role required:** admin or source_control
- **Create a dedicated GIT repository** for the application. For increased security, enable multi-factor authentication for the GIT repository.
- **Generate an access token** that the source control integration can use instead of a password and multi-factor authentication passkey. Search for personal access token on GitHub or GitLab.
- **Restrict permissions** on the access token to allow read and write access to the GIT repository.
- **Verify** that the non-production instance has network access to the GIT repository.

**Note:** The source control integration does not support linking to global applications.

The source control integration does not support linking to an application on a production instance. Instead install applications on a production instance from the application repository, an update set, or the ServiceNow Store.

1. Open the application you want to link to source control in Studio.
2. Navigate to **Source Control > Link to Source Control**.
   Studio displays the Link to Source Control dialog box.
3. Enter the connection details for the Git repository.

**Source control connection details**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>The URL to the Git repository where you want to save application files.</td>
</tr>
<tr>
<td>User name</td>
<td>The user credentials to access the Git repository.</td>
</tr>
<tr>
<td>Password</td>
<td>The password credentials or access token to access the Git repository.</td>
</tr>
<tr>
<td>Comment</td>
<td>An optional description of the repository or application.</td>
</tr>
</tbody>
</table>

**Note:** All application developers on the instance share a single set of repository credentials.

4. Click **Submit**.

The system validates the connection and user credentials and displays a success message.
All application developers on the instance can use the linked GIT repository to manage changes.

**Import application from source control**

Import an application from a source control repository to continue developing it on this instance.

- Role required: admin or source_control
- Verify that the non-production instance has network access to the GIT repository.
- Verify that the repository contains a valid application.

The source control integration does not support importing an application on a production instance. Instead install applications on a production instance from the application repository, an update set, or the ServiceNow Store.

1. Navigate to **System Applications > Studio**.
   The system displays the Welcome to Studio page.
2. Click **Open Studio > Go**.
   The system opens Studio and the Switch Applications window.
3. Click **Import from Source Control**.
   Studio displays the Import from Source Control fields.
4. Enter the following field values.

**Import from source control fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>The URL to the GIT repository where the application files reside.</td>
</tr>
<tr>
<td>User name</td>
<td>The user credentials to access the GIT repository.</td>
</tr>
<tr>
<td>Password</td>
<td>The password credentials to access the GIT repository.</td>
</tr>
</tbody>
</table>

**Note:** All application developers on the instance share a single set of repository credentials.

5. Click **Commit**.
   The system imports the application.
6. Click **Select Application**.
   Studio displays the application as a new choice in the Switch Applications window.

Select the imported application to edit it.

**Apply remote changes**

Application developers can pull changes from a linked GIT repository to apply remote changes to the local instance.

- **Role required**: admin or source_control
- **An existing link to a GIT repository**

   Navigate to **Source Control > Apply Remote Changes**.
   The following operations occur:

   - The system fetches the most recent changes from the remote repository.
   - The system applies the remote changes to the instance.
   - The system identifies any change conflicts requiring resolution.

   If there are conflicts, the system displays the **Resolve Conflicts** window.

Resolve any change conflicts.

**Commit changes**

Application developers can commit all current changes on the instance to the linked GIT repository.

- **Role required**: admin or source_control
- **An existing link to a GIT repository**

1. Navigate to **Source Control > Commit Changes**.
The system displays the Commit Changes window.

2. In **Comment**, enter a commit comment for the changes.
3. Click **Commit Changes**.
   The following operations occur:
   - The system identifies all local changes.
   - The system commits all local changes to the remote repository.

**Stash local changes**

Application developers can remove and save changes locally to apply them later.

- Role required: admin or source_control
- Link an application to source control
- Change one or more application files

Stashing changes removes them from the current application and saves them for a developer to later apply or delete.

1. From Studio, navigate to **Source Control > Stash Local Changes**.
   The system displays a list of locally changed files.
2. Click **Stash Local Changes**.
   The system saves the current changes and displays a success message.

- Close dialog
- **Manage stashes**

**Manage stashes**

Application developers can apply or delete stashed changes from Studio.

- Role required: admin or source_control
- Link an application to source control.
- Stash one or more application file changes.

1. From Studio, navigate to **Source Control > Manage Stashes**.
   The system displays a list of locally stashed changes.
2. Click the action next to the stash you want to manage.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply</td>
<td>Commits the stashed changes to the application and checks for conflicts.</td>
</tr>
<tr>
<td>Delete</td>
<td>Removes the stashed changes.</td>
</tr>
</tbody>
</table>

Resolve conflicts

Application developers can choose which application file version to use when applying remote or stashed changes.

- Role required: admin or source control
- Link an application to source control
- Apply a stashed change

Conflicts occur when there are multiple change versions of the same application file: one set of changes in the remote or stashed version and another set of changes in the local version. Studio requires developers resolve conflicts before applying remote or stashed changes.

1. From Studio, apply remote or stash changes.
   If the system identifies a conflict, it displays the Resolve Conflicts dialog.
2. Select how to resolve the conflict.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select an action</td>
<td>Apply or discard all stashed changes. Go to Step 3.</td>
</tr>
<tr>
<td>Manually merge changes</td>
<td>Individually select which changes to apply. Go to Step 6.</td>
</tr>
</tbody>
</table>

3. If you want to apply or discard all stashed changes, select an Action.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take Stashed Changes</td>
<td>Applies the application file version from the stashed changes.</td>
</tr>
<tr>
<td>Discard Stashed Changes</td>
<td>Applies the application file version from the most recent pull from the repository.</td>
</tr>
</tbody>
</table>

4. Click Apply Stashed Changes.
   The system applies the selected changes.
5. Click Close Dialog.
6. If you want to merge the conflicting changes, click Manually Apply.
   The system displays a list of version differences by field.
7. Select the field values you want the merged application file to have.
8. Click Save Merge.
   The system applies the selected changes.

Create branch

Application developers can create a branch to work on a new version of an existing application.

- Role required: admin or source_control
- An existing link to a GIT repository

1. Navigate to Source Control > Create Branch.
   Studio opens the Create Branch window.
2. Enter the **Branch Name**.
3. Click **Submit**.
   Studio creates the branch.

4. Click **OK**.
   Commit changes to the new branch.

**Switch branch**

Application developers can switch to a different repository branch to work on another version of the application.

- **Role required:** admin or source_control
- **GIT repository with one or more available branches.**

1. Navigate to **Source Control > Switch Branch**.
   The system displays the Switch Branch window.
2. Optional: If there any local changes on the instance, you can save or discard them.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save local changes</td>
<td>Saves local changes before switching to an alternate branch. You can later merge or discard the saved changes.</td>
</tr>
<tr>
<td>Discard local changes</td>
<td>Permanently deletes all local changes before switching to an alternate branch.</td>
</tr>
</tbody>
</table>

**Note:** Use caution when discarding local changes. Since all application developers share repository credentials, there is no way to discard just one set of user changes. Note you cannot later restore discarded changes.

3. Select the branch you want to switch to.
4. Click **Switch Branch**. Studio updates the local application to match the branch version from repository.

**View commit history**

Application developers can view the commit history of applications linked to a source control repository.

- Role required: admin or source_control
- An existing link to a GIT repository

1. Navigate to **Source Control > View History**. The system displays the History window.
2. Select the commit sort order type.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Sort by commit date.</td>
</tr>
<tr>
<td>Committer</td>
<td>Sort by user name.</td>
</tr>
</tbody>
</table>

3. Select the sort order direction.
## Service Creator

Service creator enables a department to offer custom services through the service catalog, such as the HR department offering tuition reimbursement for further education.

Each published service has an associated record producer catalog item. Users designated as managers and editors create and design these catalog items. End users can request services by ordering the catalog item.

All services belong to a published service category, which has an associated application and modules. When a user orders the catalog item for a service, the ServiceNow system creates a new task record within the application for that service category. Users designated as service fulfillers for the department complete these tasks to fulfill the service request.

### Service creator process

The service creator process involves requesting and publishing a service category, designating editors and service fulfillers, creating and publishing services, and submitting and fulfilling service requests.

#### Request and publish a service category

A user, typically the department manager, can request a service category for the department. This user provides high-level information regarding the service category, such as the name, the department, and the manager for the service category.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descending</td>
<td>Sort dates from the most recent to oldest date. Sort user names reverse-alphabetically from Z to A.</td>
</tr>
<tr>
<td>Ascending</td>
<td>Sort dates from the oldest to most recent date. Sort user names alphabetically from A to Z.</td>
</tr>
</tbody>
</table>

The system sorts commits by the selected sort order.

4. Select a commit.
   The system displays the commit details for the selected commit.

5. Review the commit details.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Committer</td>
<td>The user who committed the change.</td>
</tr>
<tr>
<td>Date</td>
<td>The date-time stamp of the commit.</td>
</tr>
<tr>
<td>SHA-1</td>
<td>The secure hash value identifying this commit in the repository.</td>
</tr>
<tr>
<td>Message</td>
<td>The commit message associated with this commit.</td>
</tr>
<tr>
<td>Files</td>
<td>The list of application files changed in this commit.</td>
</tr>
</tbody>
</table>
A catalog administrator can approve the request which publishes the service category, creates a ServiceNow application for managing service requests associated with the category, and creates system components for the application.

**Designate editors and service fulfilledrs**

After a service category is published, the associated manager designates editors and service fulfillers. Editors can create and modify services within that service category. Service fulfillers can complete tasks that are generated by service requests.

The manager, editors, and service fulfillers must be members of the department the service category belongs to.

**Create and publish services**

The manager and editors create services within a service category. The service design interface provides a work area for creating and modifying services.

When the service is complete, the manager publishes the service to the service catalog.

**Submit and fulfill service requests**

End users can request published services by submitting a service catalog request. This request creates a new task record within the service category application. Service fulfillers then complete the task to fulfill the service request.

**Activate Service Creator**

If the Service Creator plugin is not already activated, an administrator can activate it to access the application.

1. Navigate to *System Definition > Plugins*.
2. Find and click the plugin name.
3. On the System Plugin form, review the plugin details and then click the *Activate/Upgrade* related link.
   
   If the plugin depends on other plugins, these plugins are listed along with their activation status.
   
   If the plugin has optional features that depend on other plugins, those plugins are listed under *Some files will not be loaded because these plugins are inactive*. The optional features are not installed until the listed plugins are installed (before or after the installation of the current plugin).
4. Optional: If available, select the *Load demo data* check box.
   
   Some plugins include demo data—Sample records that are designed to illustrate plugin features for common use cases. Loading demo data is a good practice when you first activate the plugin on a development or test instance.
   
   You can also load demo data after the plugin is activated by clicking the *Load Demo Data Only* related link on the System Plugin form.
5. Click *Activate*.

**Installed with Service Creator**

Several types of components are installed with Service Creator.
Demo data is available with Service Creator. The demo data provides the Departmental Services service catalog category.

Creating a new service category also creates components for that service category.

The following components are added with Service Creator:

### Tables

**Service Creator tables**

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Category (catalog_category_request)</td>
<td>Stores all service categories.</td>
</tr>
<tr>
<td>Service Category Request User (catalog_category_request_user)</td>
<td>Tracks fulfillers for a service category. Use these records to grant or remove roles as needed.</td>
</tr>
<tr>
<td>Service (sc_cat_item_producer_service)</td>
<td>Stores all services.</td>
</tr>
<tr>
<td>Service Category App Menu (service_category_app_menu)</td>
<td>Stores the application menus for each service category.</td>
</tr>
<tr>
<td>Service Category User Role (service_category_user_role)</td>
<td>Tracks users who have been granted a role due to being an editor of a service category.</td>
</tr>
</tbody>
</table>

### UI actions

<table>
<thead>
<tr>
<th>UI action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Category and Table</td>
<td>Approves a requested service category and creates system components for that category.</td>
</tr>
<tr>
<td>Request Category Publication</td>
<td>Lets a service creator request their category be published.</td>
</tr>
<tr>
<td>Create New Service</td>
<td>Creates a new service within the service category.</td>
</tr>
<tr>
<td>View Table Definition</td>
<td>Opens the task table definition (sys_db_object) for a service category.</td>
</tr>
<tr>
<td>View Task List</td>
<td>Opens the list of tasks associated with the service category.</td>
</tr>
</tbody>
</table>

### UI policies

<table>
<thead>
<tr>
<th>UI policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hide Due Date</td>
<td>Hides the Due date field on the Service Category form if State is Requested or Due date is empty.</td>
</tr>
<tr>
<td>Hide Category If Empty</td>
<td>Hides the Category field, if empty, on the Service Category form.</td>
</tr>
<tr>
<td>UI policy</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Show Published</td>
<td>Shows the Published check box on the Service Category form if State is Created but Unpublished or Ready for Publication.</td>
</tr>
<tr>
<td>Hide Table name</td>
<td>Shows Table and hides Table name on the Service Category form if Table has a value.</td>
</tr>
<tr>
<td>Hide Category Name</td>
<td>Hides Name on the Service Category form if State is Requested or Rejected.</td>
</tr>
<tr>
<td>Table name read only</td>
<td>Makes Department and Table name read only on the Service Category form if State is not Requested.</td>
</tr>
<tr>
<td>Hide Editors</td>
<td>Hides the Editors field on the Service Category form if State is Requested or Rejected.</td>
</tr>
</tbody>
</table>

### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| glide.citizen_developer.category.auto_publish | Automatically adds new service categories to the service catalog as subcategories of the Departmental Services category.  
  - Type: true | false  
  - Default value: true  
  - Location: System Properties (sys_properties) table |
| glide.citizen_developer.set_category_roles    | Comma-separated list of roles that can set the category for a new service.  
  - Type: String  
  - Default value: admin, catalog_admin  
  - Location: System Properties (sys_properties) table |
| glide.service_creator.auto_add_to_category    | Automatically adds new services to the Departmental Services service catalog category, in addition to the department-specific category.  
  - Type: true | false  
  - Default value: true  
  - Location: System Properties (sys_properties) table |
### Script includes

<table>
<thead>
<tr>
<th>Script include</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceCategoryIsUnpublished</td>
<td>Global function that returns true if the service category is unpublished.</td>
</tr>
<tr>
<td>getMyCatalogCategories</td>
<td>Global function that returns a list of categories for which the current user is the manager or an editor.</td>
</tr>
</tbody>
</table>

### Client scripts

<table>
<thead>
<tr>
<th>Client script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplicate Category Name Check</td>
<td>Displays a warning on the Service Category Request form when the requested service category has the same name as an existing service category.</td>
</tr>
<tr>
<td>Fix Table Name</td>
<td>Ensures a valid table name on the Service Category Request form.</td>
</tr>
<tr>
<td>Hide Draft Services</td>
<td>Hides the Draft Services related list on the Service Category Request form.</td>
</tr>
<tr>
<td>Propose Table Name</td>
<td>Proposes a valid table name on the Service Category Request form for new service category requests.</td>
</tr>
<tr>
<td>Category Published</td>
<td>Displays a help message when Published is selected on the Service Category Request form.</td>
</tr>
<tr>
<td>Hide Fulfillers</td>
<td>Hides the Fulfillers related list on the Service Category Request form when appropriate.</td>
</tr>
<tr>
<td>Editors Message</td>
<td>Displays a help message for the Editors field when appropriate.</td>
</tr>
<tr>
<td>Other Tables Message</td>
<td>Provides information about existing service category tables for the selected Department.</td>
</tr>
<tr>
<td>State Message</td>
<td>Displays a help message for the State field.</td>
</tr>
</tbody>
</table>

### Business rules

<table>
<thead>
<tr>
<th>Business rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Query</td>
<td>Restricts users without the catalog_admin role to viewing service records within service categories they are the manager or editor of.</td>
</tr>
<tr>
<td>New Service</td>
<td>Provides a message when a new sc_cat_item_producer_service record is created.</td>
</tr>
<tr>
<td>Business rule</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Table Name Required</td>
<td>Ensures a service category request has a valid table name before approval.</td>
</tr>
<tr>
<td>Remove Fulfiller Role</td>
<td>Removes relevant role from service fulfillers when they are removed from a category.</td>
</tr>
<tr>
<td>Category Request query</td>
<td>Restricts users without the catalog_admin role to viewing service category records they are a manager or editor of.</td>
</tr>
<tr>
<td>Editor Role</td>
<td>Adds and removes relevant roles from service category editors.</td>
</tr>
<tr>
<td>Delete User Role</td>
<td>Removes the relevant role from service category editors when appropriate.</td>
</tr>
<tr>
<td>Category Published</td>
<td>Sets State to Published to Catalog when the Published check box is selected on the Service Category Request form.</td>
</tr>
<tr>
<td>Populate Service Name if Empty</td>
<td>Populates a service name if none is provided.</td>
</tr>
<tr>
<td>Add Departmental Services Category</td>
<td>Adds a new service to the Departmental Services service catalog category.</td>
</tr>
<tr>
<td>Default Fulfillment User</td>
<td>Makes a category manager the assignee of service tasks if no assignee is specified.</td>
</tr>
<tr>
<td>Scratchpad Draft Services Count</td>
<td>Generates field help messages.</td>
</tr>
<tr>
<td>Catalog Category Request Approved</td>
<td>Creates components necessary to use of a new service category.</td>
</tr>
<tr>
<td>Manager Role</td>
<td>Grants relevant roles to category managers.</td>
</tr>
<tr>
<td>New Service Script</td>
<td>Populates the script of a new Service to set assignment group or user.</td>
</tr>
<tr>
<td>getDepartmentUsers</td>
<td>Returns the users of a department.</td>
</tr>
<tr>
<td>Draft Item Query</td>
<td>Restricts users without the catalog_admin role to viewing draft service records they are a manager or editor of.</td>
</tr>
<tr>
<td>Grant Fulfiller Role</td>
<td>Grants relevant role to service fulfillers.</td>
</tr>
<tr>
<td>Scratchpad Department Name</td>
<td>Generates field help messages.</td>
</tr>
<tr>
<td>Scratchpad</td>
<td>Generates field help messages.</td>
</tr>
<tr>
<td>Other Tables For Department</td>
<td>Generates field help messages.</td>
</tr>
<tr>
<td>Set Single Catalog from Single Category</td>
<td>Populates a default Catalog for a new service.</td>
</tr>
</tbody>
</table>
Email notifications

Service Creator email notifications

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Category Published</td>
<td>Notifies the manager of a service category when the category request is approved.</td>
</tr>
<tr>
<td>Service Category Rejected</td>
<td>Notifies the manager of a service category when the category request is rejected.</td>
</tr>
<tr>
<td>Service Category Request Inserted</td>
<td>Notifies catalog administrators when a new category request is created.</td>
</tr>
<tr>
<td>Service Category Created</td>
<td>Notifies the manager of a service category when the category is created.</td>
</tr>
<tr>
<td>Service Category Publication Requested</td>
<td>Notifies catalog administrators when publication of a category has been requested.</td>
</tr>
<tr>
<td>Service Category Request Opened</td>
<td>Notifies the manager of a service category when a new category request is created on their behalf.</td>
</tr>
</tbody>
</table>

Components created with new service categories

When you publish a new service category using the Service Creator application, the ServiceNow system creates components for the services in that category.

These components are distinct from the components installed with the Service Creator application. The following components are added for each new service category:

Tables created with new service categories

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Department Name&gt; Tasks (&lt;service category table name&gt;)</td>
<td>The table that stores request task records for the service category. This table extends the Task table. The name of this table is defined by the department the service category is associated with, and the Table name field on the service category record. A new application menu and modules are created to allow users to access records on this table. Records on this table are numbered using a new Numbers (sys_numbers) record.</td>
</tr>
</tbody>
</table>
### User roles created with new service categories

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;service category table name&gt;_user</code></td>
<td>The user role required to access request records for a service category. The Table name for the service category determines the name of the role. Users designated as the manager, editors, or service fulfillers for a service category automatically receive this role. Only users with this role can be assigned task records for the service category. ACLs are created to allow users with this role to access the relevant service task table.</td>
</tr>
</tbody>
</table>

### Email notifications created with new service categories

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task commented for group</td>
<td>Notifies the group a service task record is assigned to when a user adds a comment.</td>
</tr>
<tr>
<td>Task commented for assignee</td>
<td>Notifies the user a service task record is assigned to when a user adds a comment.</td>
</tr>
<tr>
<td>Task closed for group</td>
<td>Notifies the group a service task record is assigned to when the record is closed.</td>
</tr>
<tr>
<td>Task worknoted for assignee</td>
<td>Notifies the user a service task record is assigned to when a user adds a work note.</td>
</tr>
<tr>
<td>Task assigned to group</td>
<td>Notifies the group a service task record is assigned to when the record is assigned to that group.</td>
</tr>
<tr>
<td>Task assigned to assignee</td>
<td>Notifies the user a service task record is assigned to when the record is assigned to that user.</td>
</tr>
<tr>
<td>Task worknoted for group</td>
<td>Notifies the group a service task record is assigned to when a user adds a work note.</td>
</tr>
<tr>
<td>Task closed for assignee</td>
<td>Notifies the user a service task record is assigned to when the record is closed.</td>
</tr>
<tr>
<td>Task opened for user</td>
<td>Notifies the user that opened a service task record when the record is created.</td>
</tr>
<tr>
<td>Task closed for user</td>
<td>Notifies the user that opened a service task record when the record is closed.</td>
</tr>
<tr>
<td>Task commented for user</td>
<td>Notifies the user that opened a service task record when a user adds a comment.</td>
</tr>
</tbody>
</table>
Forms created with new service categories

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;department name&gt; Task</td>
<td>The form for viewing request task records for the service category. By default, this form uses a layout that includes a formatter to display the questions for the service and the answers provided by the requesting user.</td>
</tr>
</tbody>
</table>

Service catalog categories

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;service category name&gt;</td>
<td>The default service catalog category for new services created within a service category.</td>
</tr>
</tbody>
</table>

Service Creator roles

The Service Creator application uses the specific roles.

Roles

<table>
<thead>
<tr>
<th>Role Title (Name)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Service Category&gt; User (&lt;service category table name&gt;_user)</td>
<td>Accesses request records for a service category. The table name for the service category determines the name of the role. Users designated as the manager, editors, or service fulfillers for a service category automatically receive this role.</td>
</tr>
<tr>
<td>Catalog Administrator [catalog_admin]</td>
<td>Creates, edits, and publishes service categories and services, and creates and edits notifications including template notifications. Catalog administrators are primarily responsible for approving service category requests.</td>
</tr>
<tr>
<td>Catalog Manager [catalog_manager]</td>
<td>Creates, edits, and publishes services, and designates editors and service fulfillers. A user designated as the manager of a service category receives this role automatically.</td>
</tr>
<tr>
<td>Catalog Editor [catalog_editor]</td>
<td>Creates and edits services. A user designated as an editor of a service category receives this role automatically.</td>
</tr>
</tbody>
</table>

Manage a service

Using the Service Creator, department managers can request a new service category, designate editors and service fulfillers for that category, and create and publish services.

Editors create and modify services. Service fulfillers complete the tasks generated from service requests.

A service category request involves assigning a service category manager, which is typically the department manager who makes the request. After the request is submitted, a catalog
To request a new service category:

1. Navigate to Self-Service > Service Catalog.
2. Select the Departmental Services category.
   
   The Departmental Services category is part of the demo data available with service creator. If this category does not exist, a catalog administrator must add the Service Category Request catalog item to an existing category.
3. Select the Service Category Request item.
4. Change the default values, as necessary (see table).
5. Click Submit.

### Managing Services

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Department</td>
<td>Department this category request is for. By default this value is the department of the current user. Changing this value also changes the Category name and Category manager values.</td>
</tr>
<tr>
<td>Category name</td>
<td>Name for the new service category. By default, ServiceNow uses a name based on the Department value.</td>
</tr>
</tbody>
</table>
### Designing services

Service creator includes an interface for designing services.

Using this interface, service category managers and editors can create and publish services, and edit service details.

All services must belong to a service category. If your department or group does not have an existing service category, you must create a new service category before you can design services for that category.

#### Add a template notification

Adding a template notification.

1. Navigate to Service Creator > Template Notifications.
2. Click New.
3. In the Send when field, select Event is fired.
4. In the Event name field, select ccrTemplate.
5. Enter other notification details.
6. Click Submit.

   The new template notification creates a notification for all service categories published after that point.

#### Notification configurations

All service categories start with a set of associated notifications, such as the notification when a task to fulfill a service request is assigned.

Notifications defined in the Service Creator > Template Notifications module are copied when a user creates a new service category.

Template notifications are distinct from the notifications for the Service Creator application itself, such as the notification when a new service category is approved or rejected. Notifications for the Service Creator application are defined in Service Creator > Notifications.

A system administrator can add and delete template notifications.

#### Create the category and table

After the request has been submitted, a catalog administrator can approve or reject the request.

Approving the request creates a new table for the service category, adds an application to the application navigator using the Category name as the application label, and sets the State of the service category to Published to Catalog.

1. Navigate to Service Creator > Category Requests.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category manager</td>
<td>Designated manager for the new service category. By default, ServiceNow uses the manager for the selected department.</td>
</tr>
<tr>
<td>Needed by</td>
<td>Date that the new service category should be available.</td>
</tr>
<tr>
<td>Comments</td>
<td>Additional comments describing the service category. This information appears as a journal entry on the Service Category form.</td>
</tr>
</tbody>
</table>
2. Open a record with a State of Requested.
3. Review the requested service category. ServiceNow provides a suggested Table name based on the Department.
   - If a service category exists with the specified category name or table name, a message appears under that field. Use a unique value for these fields.
4. Click Create Category and Table to approve the request or Reject to reject the request.
   If notifications are enabled for the instance, the service category Manager is notified of the approval or rejection.

After publishing the service category, you can access the new table by navigating to the new application in the application navigator, or by clicking the View Task List related link on the Service Category form.

Delete a template notification
Deleting a template notification prevents new service categories from using the notification, but does not delete notifications for service categories that have already been created.
1. Navigate to Service Creator > Template Notifications.
2. Select a notification record.
3. Click Delete.
4. Click OK to confirm.
**Designate an editor**

Editors can create and modify services within a service category.

Editors automatically receive the catalog_editor role.

The service category manager can designate editors for a published service category.

1. Navigate to **Service Creator > My Service Categories**.
2. Select a record with a **State** of **Published to Catalog**.
3. Click the lock icon beside the **Editors** field.
4. Select users to designate as editors using the reference lookup icon.

Only users in the appropriate department are available for selection.

5. After adding all editors, click **Update**.

Editors receive the Catalog Editor role.

**Designate a service fulfiller**

Service fulfillers can complete service requests submitted for a service category.

Service fulfillers can access applications for service categories they are assigned to, but cannot access the **Service Creator** application.
The Service Fulfillers related list on the Service Category form displays all users assigned as fulfillers for that service category. The service category manager can designate service fulfillers for a service category.

1. Navigate to Service Creator > My Service Categories.
2. Select a service category with a State of Published to Catalog.
3. In the Service Fulfillers related list, click Edit.
4. Use the slushbucket to add the appropriate service fulfillers.
   Only users in the appropriate department are available for selection.
5. Click Save.

Fulfill a service request

End users can request published services through the service catalog.

When a user requests a service, the ServiceNow system creates a new task for that service category. Service fulfillers complete these tasks to fulfill the request.

New request tasks are automatically assigned to the group or user specified in the Fulfillment Group or Fulfillment User service setting. If no fulfillment group or user is set, new records are assigned to the service category manager.

Questions for a particular service and the answers entered by the requesting user appear in the Variables section on the fulfillment task record.

1. Navigate to <Your service application> > Assigned to me.
2. Select a record.
3. Review the information presented.
4. Complete the task in accordance with department policies and procedures.
5. Set the State of the service request record to Closed Complete.
6. Click Update.
Publish a service

A service must be published to appear in the service catalog. When first created, new services appear in the Draft Services related list for the service category. Published services appear in the Services related list for the service category. The manager of a service category can publish draft services.

1. Navigate to Service Creator > My Service Categories.
2. Select a service category with a State of Published to Catalog.
3. On the Service Category form, right-click a service in the Draft Services related list.
4. Select Publish.

Team Development

Team Development supports parallel development on multiple, non-production ServiceNow instances.

Team Development provides the following features:

- Branching operations, including pushing and pulling record versions between instances.
- The ability to compare a development instance to other development instances.
- A central dashboard for all Team Development activities.

Team Development overview

Team Development allows developers to work on separate development instances while sharing code and resolving collisions throughout the development process.

After setting up the instance hierarchy, you can develop changes on your local development instance. Use the team dashboard to manage Team Development activities, such as:

- Tracking local changes and determining which changes to promote to the parent development instance.
- Pulling changes from the parent instance and resolving any collisions with local changes.
- Comparing your instance with other development instances and resolving any collisions with other development projects.
- Pushing changes when a feature is tested and ready to promote to the parent development instance.

Developers with admin access to their development instance and the parent instance can use team development. For alternative access settings, see Granting access rights to developers.

When to use Team Development

Team Development allows multiple developers to work on applications.
<table>
<thead>
<tr>
<th>Deployment option</th>
<th>Good for</th>
<th>Future considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update Sets</td>
<td>Storing changes to a baseline or installed application.</td>
<td>You can manually create update sets to store a particular application version. Use update sets to deploy patches or changes to installed applications.</td>
</tr>
<tr>
<td></td>
<td>Storing and applying a particular version of an application.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Producing a file for export.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> Do not use update sets to install applications. Instead, use the application repository or the ServiceNow Store to install applications.</td>
</tr>
<tr>
<td>Application Repository</td>
<td>Installing and updating applications on all company instances.</td>
<td>Consider uploading an application to the ServiceNow Store to share it with other users.</td>
</tr>
<tr>
<td></td>
<td>Automatically managing application update sets.</td>
<td>Allows installation of and update to the latest application version only.</td>
</tr>
<tr>
<td></td>
<td>Restricting access to applications to the same company.</td>
<td>Use update sets to store prior application versions.</td>
</tr>
<tr>
<td></td>
<td>Deploying completed applications to end users.</td>
<td><strong>Note:</strong> If used with team development, publish applications only from a parent instance.</td>
</tr>
<tr>
<td>Team Development</td>
<td>Providing change management across multiple instances.</td>
<td>Consider providing each development team access to a dedicated development instance.</td>
</tr>
<tr>
<td></td>
<td>Allowing multiple developers to work on applications.</td>
<td>Requires developers to manually merge colliding changes.</td>
</tr>
<tr>
<td></td>
<td>Organizations that have access to several non-production instances.</td>
<td>Works only for instances owned by the same organization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If used with the application repository, publish applications from a parent instance.</td>
</tr>
</tbody>
</table>

### Local changes

The Local Changes table tracks which customized records have current versions that exist on the development instance but not on the parent instance.

Use local changes to collect changes in preparation for a push.

You *queue* local changes that are ready to push. Each development instance maintains a single queue, regardless of who develops or queues the changes. You *ignore* local changes that you do not want to push. For example, you may want to ignore changes to the color scheme that visually
distinguish a development instance from the production instance. You can remove a change from the queue or stop ignoring a change.

Changing the parent instance or reconciling recreates the list of local changes that have not been queued or ignored. If you had previously queued or ignored a local change, that designation is maintained.

Local change lists

On the team dashboard, the **Local Changes** list shows the local changes that have not been queued for the next push or ignored for all pushes.

The **Ready to Push** list shows the changes that are queued, and the **Ignored** list shows the changes that are ignored. Use any of these methods to navigate a list of local changes.

**Local changes list**

<table>
<thead>
<tr>
<th>Action</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click the reference icon beside the row</td>
<td>To open the local change record itself.</td>
</tr>
<tr>
<td>Click the link in the first column</td>
<td>To open the customized record.</td>
</tr>
<tr>
<td>Right-click the row and select <strong>Show Changes Since Last Pull</strong></td>
<td>To view a comparison between the current local version and the version most recently pulled from or pushed to the parent.</td>
</tr>
<tr>
<td>Right-click the row and select <strong>Show Application File</strong></td>
<td>To open the application file for the customized record.</td>
</tr>
<tr>
<td>Right-click the row and select <strong>Show Version</strong></td>
<td>To open the current version record.</td>
</tr>
</tbody>
</table>

**Pull exceptions**

Pulling ignores version when certain conditions occur.

**Pull exceptions table**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matched an exclusion policy</td>
<td>An exclusion policy prevents pulling changes for records matching the policy conditions. The pull identifies the changes but does not include versions for these records.</td>
</tr>
<tr>
<td>Private properties</td>
<td>A private property is excluded from all Update Sets and pulls.</td>
</tr>
<tr>
<td>Collisions</td>
<td>A collision is detected when the pulled version and the current local version both include modifications to the same record. You must resolve all collisions before you can pull.</td>
</tr>
<tr>
<td>Previously resolved collisions</td>
<td>When you resolved a collision by accepting either the pulled version or local version of a record, the pull remembers your decision and accepts the version that you indicated as a “previously resolved collision”.</td>
</tr>
</tbody>
</table>
### Team dashboard

The team dashboard provides a central place to manage all Team Development activities on your development instance.

You can track local changes, pull and push changes between the local and parent instances, compare the local instance to other development instances, and resolve any collisions. You can also reconcile with the current parent instance or change the parent instance.

To access the dashboard, navigate to **Team Development > Team Dashboard**.
The control panel in the top left provides status indicators and Team Development actions.

- **Parent:** indicates the status of the connection to the parent instance. If a problem or warning is detected, point to the indicator to view the error messages, or click the indicator to open the remote instance record.
- **Change:** changes the parent instance. See *Changing the Parent Instance.*
- **Reconcile:** compares the development instance to the parent instance. See *Reconciling.*
- **Ready to Pull:** indicates the number of changes on the parent that have not been pulled to the local instance.
- **Pull:** initiates a pull. See *Pulling Versions.*
- **Push:** opens a page that allows you to review the changes before a push. See *Pushing Versions.*
- **Refresh:** updates the status indicators on the control panel. The dashboard updates only when you reload or refresh the page.
- **Local:** indicates the status of the most recent comparison with another instance. If collisions are detected, click the indicator to open the list and resolve the collisions. See *Resolve a collision in Team Development.*
- **Collisions:** appears only if any local changes collide with versions pulled from the parent and indicates the number of collisions. Click the indicator to open the list and resolve the collisions. See *Resolve a collision in Team Development.*
- **Compare to:** allows you to select another development instance to compare with the local instance. See *Comparing to Peer Instances.*
- **Ready to Push:** indicates the number of local changes that are queued for the next push. See *Queuing and Ignoring Local Changes.*
- **Local changes:** indicates the number of local changes that have not been queued or ignored. Click the indicator to open a list of these changes.
- **Ignored:** appears only if any local changes are ignored and indicates the number of ignored changes. Click the indicator to open a list of these changes.

The team dashboard includes lists for tracking local changes and viewing the history of Team Development activities.

- **Local changes:** lists the local changes that have not been queued or ignored.
- **Pushes and Pulls:** provides a history of pushes and pulls. Expand a row to see the customized records for which versions were transferred as part of the push or pull.
- **Instance Comparisons:** provides a history of comparisons with other development instances.
- **Collisions:** lists the collisions that must be resolved before the next pull or push. You can right-click a row and select *Resolve Collision.* See *Resolving Collisions.*
- **Ready to Push:** lists the local changes that have been queued for the next push.
- **Ignored:** lists the local changes that are ignored for all pushes.

### Approve or reject a push

Code reviewers must approve or reject a push from the Team Development application.

Although reviewers can see the individual versions within a push, they must approve or reject the push as a whole.

1. Log in to the parent instance that requires code review.
2. Navigate to **Team Development > Code Review Requests.**
3. Select a change in the **Awaiting Code Review** stage.
4. Review the changes in the **Push or Pull Versions** related list.
5. Click **Approve or Reject.**
6. Optional: Enter review comments in **Comments.** These comments are visible to anyone who can see the Pushes and Pulls history.
7. Click either **Approve** or **Reject**, as appropriate.

![Approve or Reject](image)

**Note:** The **URL** and **Remote Instance** fields list the address and name of the instance where the change originated.

### Back out a local change

Back out all local changes and restore the last version reconciled with the parent instance.

1. **Define a parent instance.**
2. **Pull changes from the parent instance.**
3. Navigate to **Team Development > Team Dashboard.**
4. Filter the **Local Changes** list to show only the changes that you want to back out.
5. Do one of the following actions:
   - Click **Back Out All.**
   - Right-click the local change you want to back out, and then click **Back Out.**
Cancel a code review request

Developers can cancel any push they submitted that is in the **Awaiting Code Review** stage.

Canceling a request sets the push to the **Code Review Request Cancelled** stage on the submitting instance. The submitting instance retains a version history of the push but the parent instance does not.

1. Log in to the instance that pushed the changes.
2. Navigate to **Team Development > Pushes and Pulls**.
3. Filter for the push you want to cancel.
Note: You cannot cancel a push that has been approved or rejected.

4. Select the Push or Pull record.
5. Click Cancel Code Review.

Change the parent instance

If it becomes necessary to modify the instance hierarchy, you can change the parent for a development instance.

Changing the parent initiates a complete comparison between the development instance and the new parent instance. To optimize comparison speed and reduce the number of collisions and local changes that need review afterwards, ensure that the new parent instance was cloned recently from an appropriate instance (for example, the production instance). Before you change the parent instance, ensure that the change does not conflict with your change management process or other development efforts.

To change the parent for a development instance:

1. On the development instance, navigate to Team Development > Team Dashboard.
2. In the control panel, click Change.
3. Select the remote instance you want to use as the parent and click Select.

Alternatively, click the link to define a new remote instance. Then, repeat steps 1–3 and select the remote instance you defined.

The system initiates a reconcile, which compares the local instance to the parent, and then generates the list of local changes and calculates the number of changes that are ready to pull from the parent.

4. On the completion page, click Team Dashboard.
5. Pull versions from the parent instance and resolve any collisions.
6. Review the local changes list and queue or ignore changes, as appropriate.

Check the review status of a pushed change

If the parent instance requires pushed changes to undergo code review, changes are placed in the Awaiting Code Review stage.

If you configure the parent instance to send notifications, it sends the submitting developer a notification when the pushed changes are approved or rejected. Developers can also manually check the status of their pushed changes from the Pushes and Pulls module on the submitting instance.

1. Log in to the instance that submitted code for review.
2. Navigate to Team Development > Pushes and Pulls.
3. Filter for the push you want to review.
   - Pushes in the **Complete** stage are approved and applied to the parent instance.
   - Pushes in the **Collided** stage are rejected because of a collision.
   - Pushes in the **Awaiting Code Review** stage are awaiting review.
   - Pushes in the **Code Changes Rejected** stage are rejected by a reviewer.
   - Pushes in the **Code Review Request Canceled** stage are canceled by the submitting developer.

4. Click the **Reviews** related list to see the following information.
   - Who submitted a review decision.
   - What the decision was: either approved or rejected
   - What comments if any the reviewer provided.
<table>
<thead>
<tr>
<th>Name</th>
<th>Test Push 001</th>
<th>Type</th>
<th>Push</th>
<th>Created</th>
<th>Created by</th>
<th>Comments:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote Instance</td>
<td>8381</td>
<td>Latest version date</td>
<td>2014-01-24 11:15:17</td>
<td>Stage</td>
<td>Compress</td>
<td></td>
</tr>
<tr>
<td>Created by</td>
<td>admin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please review my changes.

### Related Links

Team Dashboard

<table>
<thead>
<tr>
<th>Push and Pull Versions (1)</th>
<th>Reviews (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push or Pull = Test Push 001</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Approver Name</th>
<th>Approver ID</th>
<th>Decision</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Admin</td>
<td>admin</td>
<td>approved</td>
<td>Looks good. Your changes are approved.</td>
</tr>
</tbody>
</table>

Actions on selected rows...
Compare a pushed version to a local version

Code reviewers can compare the pushed versions to the local versions to see the potential effect of incoming changes.

1. Log in to the instance requiring code review.
2. Navigate to Team Development > Code Review Requests.
3. Select a change in the Awaiting Code Review stage.
4. Review the changes in the Push or Pull Versions related list.
5. Right-click a row in the list and click Compare to Current. A comparison of the differences between the pushed and local versions appears.

Compare to peer instances

You can compare the local instance to any other remote instance and commit any current versions from the remote instance on your development instance.

Comparing allows you to share code between instances without pushing to a common parent.

Comparing instances does not automatically commit any versions on the local instance. It initiates a full comparison of all changes on the remote instance and all changes on the local instance, and then reports which customized records have different current versions. You can selectively commit a version from the remote instance or compare it with the version on your local instance. You can delete the instance comparison record when you finish evaluating the differences.

To compare the local instance to a peer instance:

1. Ensure that the peer instance is defined as a remote instance.
2. Navigate to Team Development > Team Dashboard.
3. In the control panel, click Compare to.
4. Select the peer instance you want to compare to the local instance and click Compare.
5. On the completion page, click Show Results. The instance comparison record opens.
6. Review the **On Remote and not Local** related list, which shows the customized records where the current version on the peer instance is not on the local instance. For each customized record, you can:

- Compare the current remote version to the current local version by right-clicking a row and selecting **Compare to Current**.
- Load the current remote version as the current local version by right-clicking a row and selecting **Load This Change**.

**Ignore a local change**

Ignoring a local change prevents updates to a record from generating new versions in the Local Changes list.

An ignored local change always points to the current version for the record. You cannot push ignored records to another instance.

**Local change action list**

<table>
<thead>
<tr>
<th>Action</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore a record that has a version queued for push</td>
<td>The queued change is deleted</td>
</tr>
<tr>
<td>Ignore a record that has a version queued for code review</td>
<td>The queued change is deleted</td>
</tr>
<tr>
<td>Pull changes for an ignored record</td>
<td>Collision</td>
</tr>
<tr>
<td>Action</td>
<td>Result</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Resolve a collision by taking the parent version</td>
<td>There is no longer a local change to ignore</td>
</tr>
<tr>
<td>Resolve a collision by keeping the local version</td>
<td>The ignored change remains on the local instance</td>
</tr>
</tbody>
</table>

1. Navigate to **Team Development > Team Dashboard**.
2. Filter the **Local Changes** list to show only the changes that you want to ignore.
   For example, filter the list to show all changes in the **Default** Update Set.
3. Click **Ignore All**.
4. (Recommended) Review the **Ignored** list to ensure that the correct changes are ignored.
### Option Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To stop ignoring changes</td>
<td>Select the check boxes beside the rows and select <strong>Do Not Ignore</strong> from the Actions choice list.</td>
</tr>
<tr>
<td>To stop ignoring changes and add them to the queue instead</td>
<td>Select the check boxes beside the rows and select <strong>Queue for Push</strong> from the Actions choice list.</td>
</tr>
</tbody>
</table>

#### Pull a version

Pulling retrieves versions of customized records from the parent instance and adds them on the development instance. Pulling does not retrieve any versions for changes made by system upgrades, but it retrieves all versions for changes made by users, not just the current version.

Pulling retrieves all versions for changes made by users that have not already be pulled onto the development instance, and you cannot choose which versions to pull. The first time you pull from a parent instance, the pull retrieves all versions for changes made by users. Subsequent pulls retrieve the new versions since your last pull. Each pull is recorded in the Push or Pull [sys_sync_history] table on the development instance. Historical versions are saved with a state of **History**.

1. Navigate to **Team Development > Team Dashboard**.
2. In the control panel, click **Pull**.
3. On the completion page, click **Show Results**. The Push or Pull form opens.

The **Push and Pull Versions** related list shows the customized records for which versions were retrieved and indicates if any pull exceptions exist.
4. Resolve any collisions.

Push a version

Pushing promotes changes from the development instance to the parent instance and commits the current version of a customized record on the development instance as the current version on the parent instance.

Pushing adds only the current development version to the parent, not all the development versions.

Note: Updates to records from different applications cannot be pushed/pulled in the same push/pull. To resolve the error in the case that updates to other applications are mixed in: De-queue the updates to other applications. Push for one application. Re-queue the updates to one application. Push and then repeat as needed.

Pushing creates a local Update Set on the parent that is marked as complete. Pushed changes are also tracked as local changes on the parent. Therefore, you can promote changes through your development and test hierarchy by transferring the Update Set or by pushing the local changes. Each push is recorded in the Push or Pull table on the development instance.

1. Navigate to Team Development > Team Dashboard.
2. Queue the local changes that are ready to push.
3. Pull versions from the parent instance and resolve any collisions.
You cannot push changes to the parent instance if collisions are detected.

4. In the control panel, click **Push**. The Push Changes page opens.

5. Provide a **Name** for the changes.

6. Review the list of changes to ensure that the correct changes are included.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To remove changes that you do not want to push</td>
<td>Select the check boxes beside the rows and select <strong>Do Not Push</strong> from the <strong>Actions</strong> choice list</td>
</tr>
<tr>
<td>To add changes</td>
<td>Click <strong>Cancel</strong> and repeat the procedure from step 2</td>
</tr>
</tbody>
</table>

7. Optional: Edit the name. The name identifies the push record on the development instance and the local Update Set record on the parent instance.

8. Optional: Enter comments. The comments are added to the push record on the development instance and the local Update Set record on the parent instance.

9. Click **Push Changes**. The system initiates a pull to ensure that there are no collisions before the push proceeds.
• If collisions are detected, the push is automatically canceled and you must repeat the procedure from step 3.
• If no collisions are detected, the changes are staged on the parent instance. On the parent, each version is validated and then committed in the correct order to maintain dependencies between records. For example, a new table is committed before a field on that table to ensure the field is properly created.

**Note:** You cannot push if there is a version conflict between instances or the pushing instance has changes in the **Awaiting Code Review** stage.

10. On the completion page, click **Show Results**.

11. Review the push record for any errors or skipped changes.
   - Changes with a state of **Pushed** were committed on the parent instance.
   - Changes with a state of **Skipped** were not committed on the parent instance and remain queued as local changes on the development instance.

12. For each skipped change, review the log message to determine why the change was skipped. Develop any changes that are necessary to commit the desired version on the parent instance, and then push them. Some examples of why a change may be skipped include:
   - A table does not exist on the parent because it was created when you activated a plugin on the development instance. Ensure the plugin is activated on the parent and push the change again.
   - An error occurred during the push. Try to push again.
   - The current version is invalid. Revert to a previous version and make the change again to ensure the version is valid.
   - An error occurred on the parent during the push. The **Log** field on the push record contains the exception message. Review the system logs on the parent instance and troubleshoot any problems with the instance.
Back out a push

Application developers can back out a push to remove unwanted changes.

1. Navigate to **Team Development > Pushes and Pulls**.
2. Select the push to back out.
3. Click **Back Out**.
4. Click **OK** when the confirmation message appears.

Queue a local change for a push

Application developers can queue a local change for a push to ensure the changes are available to other developers.

1. Navigate to **Team Development > Team Dashboard**.
2. Filter the **Local Changes** list to show only the changes that are ready to push.
   - For example, filter the list to show only the changes associated with a particular application.
3. Click **Queue All For Push**.

4. (Recommended) Review the **Ready to Push** list to ensure that the correct changes are in the queue.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To remove changes from the queue</td>
<td>Select the check boxes beside the rows and select <strong>Do Not Push</strong> from the <strong>Actions</strong> choice list.</td>
</tr>
<tr>
<td>To remove changes from the queue and choose to ignore them instead</td>
<td>Select the check boxes beside the rows and select <strong>Ignore This Change</strong> from the <strong>Actions</strong> choice list.</td>
</tr>
</tbody>
</table>
Reconcile changes

Reconciling first compares the local instance to the parent, and then generates the list of local changes and calculates the number of changes that are ready to pull from the parent.

A reconcile occurs automatically whenever you select a parent instance. You may need to manually reconcile after an external disruptive event on the parent instance, such as a clone or failover.

Note: This process may take a while to complete depending on the size and age of the instance.

1. Navigate to Team Development > Team Dashboard.
2. In the control panel, click Reconcile.
3. In the confirmation dialog box, click OK.

The list of local changes that have not been queued or ignored is recreated. If you had previously queued or ignored a local change, that designation is maintained.

4. Optional: On the completion page, click Show Results. Review the instance comparison record.
   - The On Remote and not Local related list shows the versions that are ready to pull from the parent.
   - The On Local and not on Remote related list shows the local versions that are ready to queue or ignore.
5. Click Team Dashboard.
6. Pull versions from the parent instance and then resolve any collisions.
7. Review the local changes list and queue or ignore changes, as appropriate.

Resolve a collision in Team Development

A collision is detected when the pulled version and the current local version are modifications of a different version, indicating that someone else has modified the same record that you have modified. The team dashboard displays the number of collisions between the local and the parent instance.

To ensure that your changes do not conflict with other development efforts, you should resolve collisions as soon as they are identified. You must resolve all collisions before you can pull or push.

1. Navigate to Team Development > Team Dashboard.
2. In the control panel, click Collisions or click the count of collisions. A list of collisions opens.
3. Right-click a row and select Resolve Collision. (Alternatively, open the record and click the Resolve Collision related link.) The Resolve Collision page displays a comparison between the version that was pulled from the parent and your local record. The page highlights the differences.
4. Review the differences. You have the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>To maintain the local record as the current version</td>
<td>Click Use Local Version. The pulled version is added to the version history for the record.</td>
</tr>
</tbody>
</table>
To load the version pulled from the parent as the current version

To move a setting from the selected version to update the current version

Note: Some types of record do not support this method. See Limitations on updating records for more information.

The system performs that action and also clears the collision for future push/pulls.

5. Repeat the process for every remaining collision.

The system saves the merged changes and resolves the collision.

Limitations on updating records

There are some types of records that you cannot merge while resolving differences on the Compare to Current and Resolve Collision pages.

Record types that allow a choice only between reverting or accepting the pulled or current record

The following record types do not allow you to merge individual values. Instead, differences involving the following record types display a read-only comparison and allow a choice between updating and reverting:

- sys_choice (Choice)
- sys_choice_set (Choice Set)
- sys_ui_form (Form)
- sys_ui_list (List)
- sys_ui_related_list (Related List)
- sys_ui_section (Form Section)
- wf_workflow (Workflow)
- wf_workflow_version (Workflow Version)

Team Development, Resolve Collision page: Use Pulled Version and Use Local Version options.

Upgrade History, Compare to Current page: Comparing non-current update versions to current update version. Allows only Revert to Base System option.

Field types that do not support merging

The following field types do not support individual merging between versions or updates:

- auto_increment (Auto Increment)
- auto_number (Auto Number)
- breakdown_element (Breakdown Element)
- catalog_preview (Catalog Preview)
- collection (Collection)
- color_display (Color Display)
- composite_field (Composite Field)
- compressed (Compressed)
- counter (Counter)
- currency (Counter)
- data_array (Data Array)
- data_object (Data Object)
- data_structure (Data Structure)
- date (Other Date)
- datetime (Basic Date/Time)
- days_of_week (Days of Week)
- document_id (Document ID)
- due_date (Due Date)
- Email (Email)
- external_names (External Names)
- field_list (Field List)
- float (Floating Point Number)
- glide_action_list (UI Action List)
- glide_precise_time (Precise Time)
- glide_var (Glide Var)
- image (Basic Image)
- index_name (Index Name)
- int (Integer String)
- integer_time (Integer Time)
- ip_address (IP Address)
- journal (Journal)
- journal_input (Journal Input)
- journal_list (Journal List)
- long (Long Integer String)
- mask_code (Mask Code)
- metric_absolute (Metric Absolute)
- metric_counter (Metric Counter)
- metric_derive (Metric Derive)
- metric_gauge (Metric Gauge)
- mid_config (MID Server Configuration)
- month_of_year (Month of Year)
- multi_small (Multiple Line Small Text Area)
- name_values (Name/Values)
- nl_task_int1 (NL Task Integer 1)
- order_index (Order Index)
- password (Password (1 Way Encrypted))
- percent_complete (Percent Complete)
- ph_number (Phone Number)
- phone_number (Phone Number (Unused))
- phone_number_e164 (Phone Number (E164))
- price (Price)
- reference_name (Reference Name)
- related_tags (Related Tags)
- reminder_field_name (Reminder Field Name)
- repeat_count (Repeat Count)
- repeat_type (Repeat Type)
- replication_payload (Replication Payload)
- schedule_date_time (Schedule Date/Time)
Resolve multiple collisions

You can resolve multiple collisions without reviewing the differences between the local and pulled versions.

1. Navigate to Team Development > Team Dashboard.
2. In the control panel, click the number of collisions. A list of collisions opens.
3. Select the check boxes beside the rows you want to resolve.
4. In the Actions choice list, use one of the following methods to resolve the collision:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To load the version pulled from the parent as the current version for all selected collisions</td>
<td>Select Use Pulled Version</td>
</tr>
<tr>
<td>To maintain the local version (local record) as the current version for all selected collisions. The pulled versions are added to the version history for the records.</td>
<td>Select Use Local Version</td>
</tr>
</tbody>
</table>

Team Development setup

To enable parallel development on multiple non-production instances, administrators can set up the Team Development instance hierarchy and grant access rights for developers.
Access rights for developers

To use Team Development, application developers must have a set of credentials for each instance in the Team Development hierarchy. An instance’s placement in the \textit{team development hierarchy} determines the credentials it requires.

\textbf{Credentials for Team Development access}

<table>
<thead>
<tr>
<th>Desired Access</th>
<th>Credential Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access to the Team Development application</td>
<td>A user with the admin role on the instance you are accessing</td>
</tr>
<tr>
<td>Right to \textit{register a remote instance}</td>
<td>One of following:\</td>
</tr>
<tr>
<td></td>
<td>• A user with the admin role on the instance you are registering</td>
</tr>
<tr>
<td></td>
<td>• A user with the teamdev_user role on the instance you are registering</td>
</tr>
<tr>
<td>Right to push changes to the parent instance from a</td>
<td>One of following:\</td>
</tr>
<tr>
<td>development instance</td>
<td>• A user with the admin role on the parent instance</td>
</tr>
<tr>
<td></td>
<td>• A user with the teamdev_user role on the parent instance</td>
</tr>
<tr>
<td>Right to compare to a registered remote instance</td>
<td>One of following:\</td>
</tr>
<tr>
<td></td>
<td>• A user with the admin role on the registered development instance</td>
</tr>
<tr>
<td></td>
<td>• A user with the teamdev_user role on the registered development instance</td>
</tr>
<tr>
<td>Access to the \textit{Code Review Requests} module</td>
<td>One of following:\</td>
</tr>
<tr>
<td></td>
<td>• A user with the admin role on the parent instance</td>
</tr>
<tr>
<td></td>
<td>• A user with the teamdev_code_reviewer role on the parent instance</td>
</tr>
</tbody>
</table>

\textbf{Note:} The teamdev\_user role does not grant access to the Team Development application and is not intended for developers to work on local development instances. It is intended to grant developers non-admin access to remote instances such as the parent instance or a peer development instance.

Create an exclusion policy

Application developers can create an exclusion policy to prevent pushes or pulls to particular instances in the team development hierarchy.

1. Navigate to \textbf{Team Development > Exclusion Policy}.
2. Click \textbf{New}.
3. Complete the Exclusion Policy form (see table).
4. Click **Submit**.

**Exclusion policy form**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a unique description of the policy.</td>
</tr>
<tr>
<td>Policy</td>
<td>Select when the policy applies. Options include:</td>
</tr>
<tr>
<td></td>
<td>· Push only</td>
</tr>
<tr>
<td></td>
<td>· Push and Pull</td>
</tr>
<tr>
<td></td>
<td>· Pull only</td>
</tr>
<tr>
<td>Remote Instance</td>
<td>(Optional) Select a specific remote instance to ignore changes from during pull operations. Leaving this field blank ignores changes from all remote instances.</td>
</tr>
<tr>
<td>Table</td>
<td>Select which table to ignore changes for.</td>
</tr>
<tr>
<td>Conditions</td>
<td>Select any additional criteria a change must meet to be ignored other than the table name. This field is only visible when the <strong>Policy</strong> is <strong>Push only</strong>.</td>
</tr>
</tbody>
</table>

**Define a remote instance**

For each instance, define other instances in the hierarchy as remote instances.

For example, to set up remote instances for Sub-Dev 1:

1. If IP address access control is enabled, log in to the remote instance and add Sub-Dev 1 as an exception.
2. On Sub-Dev 1, navigate to **Team Development > Remote Instances**.
3. Click **New**.
4. Define the remote instance, such as Dev-Parent, by completing the form (see table).
5. Click **Submit**.
6. Repeat *step 1 through step 5* for each instance in the hierarchy that this instance needs to push and pull with (for example, Sub-Dev 2 and Sub-Dev 3).

### Remote instance form

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a unique name describing the instance.</td>
</tr>
<tr>
<td>Type</td>
<td>Specify whether the remote instance is a development, test, or UAT instance.</td>
</tr>
<tr>
<td>Active</td>
<td>Specify whether the local instance communicates with the remote instance as a member of Team Development. Team Development operations such as comparing changes between instances or selecting a parent instance are only available for active remote instances.</td>
</tr>
<tr>
<td>URL</td>
<td>Specify the URL of the remote instance using the appropriate transfer protocol. Each remote instance record should have a unique URL. Creating duplicate records with the same URL can cause errors.</td>
</tr>
<tr>
<td>Username</td>
<td>Enter the user on the remote instance who authorizes Team Development operations on the instance. This user account must have an appropriate role on the remote instance.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the password of the authorizing user.</td>
</tr>
</tbody>
</table>
Enable a code review

You can require a code review of all changes pushed to an instance.

1. Navigate to Team Development > Properties.
2. Select the Yes check box for If this property is set to Yes, code review is required before pushing to this instance (com.snc.teamdev.requires_codereview).
3. Click Save.

Setting this property adds the Code Review Requests module to the application menu and requires all changes pushed to this instance to remain in the Awaiting Code Review stage until someone in the Team Development Code Reviewers group approves them.

Select the parent instance

An instance can have multiple peer instances but only one parent instance.

The parent instance is the only instance you can pull changes from and push changes to. The parent instance must be on the same release family as the local instance. For example, a development instance on the Geneva release family must have a parent instance also on the Geneva release family. If you select a parent from a different release family, the Team Development dashboard displays an error message and prevents you from pulling changes and reconciling. If you select a parent from a different patch release, the dashboard displays a warning message but allows you to pull changes and reconcile.

Do not use Team Development with production or test instances.
- Do not use a test or production instance as the parent instance in Team Development.
- Do not make any instance the parent of a production instance.
- Production instances should never have a parent.

When you back out a change on a Team Development instance, it backs out the change all the way back down the chain, including undoing the work on the source instance. This behavior can cause major problems on test and production instances.

1. Navigate to Team Development > Team Dashboard.
2. In the control panel, click the appropriate link:

<table>
<thead>
<tr>
<th>Team dashboard control panel options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use &lt;instance name and URL&gt;</td>
<td>Selects the most recently defined remote instance as the parent instance.</td>
</tr>
<tr>
<td>Select a different instance</td>
<td>Opens a dialog box where you can select another remote instance or define a new remote instance.</td>
</tr>
<tr>
<td>Register a new instance or List all remote instances</td>
<td>Opens the remote instance form or list, where you can define a new remote instance. These options are available when no remote instances are defined.</td>
</tr>
</tbody>
</table>
3. If you defined a new remote instance in step 2, repeat step 1 through step 2 and select the remote instance you defined.

The system initiates a reconcile, which compares the local instance to the parent. It then generates the list of local changes and calculates the number of changes that are ready to pull from the parent. The reconcile also validates the instance versions.

4. Pull all changes from the parent instance if both instances are in the same release family.

   **Note:** The parent instance is saved in the `glide.apps.hub.current` system property.

### Set up an instance hierarchy

Set up an instance hierarchy that best supports your development life cycle.

This example demonstrates how to set up an instance hierarchy where several peer sub-development instances have the same parent development instance, but a more complex configuration may be required to handle multiple project teams or other customer requirements.

Do not use Team Development with production or test instances.

- Do not use a test or production instance as the parent instance in Team Development.
- Do not make any instance the parent of a production instance.
- Production instances should never have a parent.

When you back out a change on a Team Development instance, it backs out the change all the way back down the chain, including undoing the work on the source instance. This behavior can cause major problems on test and production instances.

1. Provision a parent development instance on the same software version, such as Dublin, as the target instance, such as production.
2. (Recommended) Clone the production instance to the parent development instance.
3. Provision sub-development instances on the same software version as the parent development instance.
4. Optional: Log in to the parent development instance and clone it to the sub-development instances.
5. On each sub-development instance:
   1. Define remote instance connections to other instances in the hierarchy that this instance needs to push and pull with.
   2. Select the parent instance.
   3. Pull all changes from the parent instance.
4. *Grant access rights* to appropriate developers.

**Code reviews**

Team Development administrators can require that pushes undergo code review before accepting pushes.

When code review is enabled, pushing a change to the parent instance triggers the code review workflow. By default, users with the `teamdev_code_reviewer` role receive notifications to review changes and can approve or reject changes. The Team Development Code Reviewers has the `teamdev_code_reviewer` role.
For each change, reviewers can see the following information.
- Which remote instance the pushed change comes from.
- Who pushed the change to the parent.
- What the change is called.
- When the change was created.
- Which versions the change includes.

Reviewers must approve or reject a push from the Team Development application.
While changes are being reviewed on the parent instance, a child instance cannot do the following activities involving the parent instance:
- Push changes to the parent instance.
- Pull changes from the parent instance.
- Reconcile changes with the parent instance.
- Change the parent instance to another instance.
- Delete the remote instance record for the parent instance.

**Code review notifications**

You must enable email notifications on the instance requiring code review for that instance to send code review notifications.

The Team Development Code Review workflow sends notifications to members of the Team Development Code Reviewers group when:
- A push requires code review.
- A user cancels a push.

If the user who pushed the changes has a user record with an email address on the instance where code review was required, the user receives a notification when the approval stage is set to Complete (approved) or Code Changes Rejected.

The code review notifications contain the following information:

<table>
<thead>
<tr>
<th>Notification name</th>
<th>Table</th>
<th>Contents</th>
</tr>
</thead>
</table>
| Code review update for developer                       | Push or Pull (sys_sync_history) | • The push name  
• The approval stage of the push (approved or rejected)  
• A link to the instance where the code review request was made |
| Notify code reviewer of canceled review                | Push or Pull (sys_sync_history) | • The user who canceled review  
• The push that was canceled                            |
- Starts when changes are pushed to the parent instance.
- Verifies that the code review property is active on the parent instance.
- Sets the stage of changes requiring approval to **Awaiting Code Review**.
- **Notifies** the Team Development Code Reviewers group to review pushed changes, if configured.
- Loads approved changes or sets the stage to **Code Changes Rejected**.

**Warning:** Use caution when modifying this workflow, as the code review feature may not function properly.

**Exclusion policies**

You can exclude certain files from change tracking by creating an exclusion policy.

When a change matches an exclusion policy, the change does not generate records in the local changes list. The change still generates local version records and Update Set records as normal. See **Creating an Exclusion Policy**.

**Note:** The exclusion policy applies to changes identified during a reconciliation operation. If you create an exclusion policy after a reconciliation, Team Development still tracks the changes until the next reconciliation.
**Instance hierarchies**

Team Development allows you to set up a distributed version control system between multiple ServiceNow instances where each instance acts as a source repository, or branch. Developers use separate instances to work on different features, applications, or product releases at the same time. With Team Development, developers can share code between these instances and resolve collisions throughout the development process.

Team Development allows you to establish hierarchical relationships between instances and provides a mechanism for transferring changes between instances that integrates with the Update Set process where necessary. In a Team Development instance hierarchy, each non-production instance has a parent instance. Instances that have the same parent instance are peer instances. The shared parent instance becomes the central hub, or repository, and all peer instances synchronize to it.

**Pulls and pushes**

Developers synchronize their instances to the parent instance by pulling and pushing versions of customized records and resolving collisions between versions on the parent instance and the development instance.

Developers can compare peer instances to one another and share code or resolve collisions before pushing versions to the parent instance.

Pulling from the parent retrieves versions of records that have customer updates. Pulling retrieves all versions that have not already been pulled onto the development instance, including historical versions, and you cannot choose which versions to pull. You must resolve any collisions before proceeding with further pulls or pushes.

Pushing to the parent adds only the current development version to the parent, not all the development versions. You can choose which changes to push to the parent. Pushing creates a local Update Set on the parent that is marked as complete. Pushed versions are also tracked as local changes on the parent. Therefore, you can promote changes through your development and test hierarchy by transferring the Update Set or by pushing the local changes.

Comparing reports the differences between two peer instances. You can choose which versions to pull from a peer instance.

The Pushes and Pulls related list on the team dashboard displays the user who created a change and the remote instance where the change was created.

**Team Development process**

The basic Team Development process sets up the instance hierarchy, grants developer access rights, manages the movement of development changes from development instances to test instances, and promotes applications to the production instance.

1. Set up the development instance hierarchy as described in *Set up an instance hierarchy*.
   1. Provision development instances on the same software version as the target instance. For example, use the software version that is running on your production instance.
   2. (Recommended) Clone the target to the development instances.
   3. For each instance, define the parent instance.
   4. (Optional) For each instance, define the peer instances.
   5. For each instance, pull all changes from the parent instance.

2. For sub-development instances, grant access rights to appropriate developers.
3. Develop customizations on sub-development instances. Use the team dashboard to track development activities.
   - Pull versions from the parent instance, such as versions that were pushed from other sub-development instances. Reconcile any conflicts with the current local version, as necessary.
   - Track local changes. Queue changes that are ready to push to the parent development instance.
   - Compare versions on peer instances. Reconcile any conflicts.

4. When a feature is ready to promote to the parent development instance, push the current version of the customized records.

5. (Optional) Have code reviewers approve or reject the pushed version.

6. Test and promote the feature into production according to your testing and release management process.

Team Development roles

To use Team Development, developers must have admin access to their development instance.

To allow pushes to the parent instance, a remote instance connection must be defined with a user account that has admin access to the parent instance.

To limit developer access to the parent instance, see Granting Access Rights to Developers.

To use code review features, users must have the teamdev_code_reviewer role. See Code Review.

Versions

Version records track changes to a customized record over time so that administrators can compare or revert to specific versions later.

Administrators can also transfer versions between instances with Update Sets or team development.

Version record navigation

There are a variety of methods for viewing a list of versions for an object.

- For forms, right-click the header and select Configure > Form Layout.
- For lists, perform the appropriate action for the list version.
  - List v2: Right-click the header and select Configure > List Layout.
  - List v3: Open the list title menu and select List Layout.
- Click the Show Versions related link.
- For tables that use the update_synch attribute, add the Versions related list to the form. This list is on several forms by default, including, business rules, UI actions, and client scripts.
- For any customizable object, right-click the form header and select Show Application File, then scroll down to the Related Record Versions related list.

You can navigate from a version record to:

- The customized object: Click the Show Related Record related link.
- The application file record for the object: Click the Show Application File related link.
Versions transferring

Administrators transfer version records between instances by moving customizations with Update Sets or the Team Development application.

- **Update sets**: committing an Update Set adds versions. For each update in the Update Set, the version that corresponds to the update is added on the local instance.
- **Team Development**:
  - **Pulling** retrieves from the parent instance all versions of customized records that have not already been pulled and then adds them on the local instance.
  - **Pushing** adds to the parent instance only the current local version, not all the local versions.
  - **Loading changes from peer instances** adds selected versions to the local instance.
Version records

The Update Versions [sys_update_version] table contains records that represent the state of a customizable object at a particular time.

A customizable record is any object that is tracked by Update Sets, such as business rules or script includes. A new version record is created automatically whenever a user changes a customizable record or changes the application file for the customizable record.

A record represents the version of a base system object as it was delivered in the most recent upgrade. Baseline versions are created only for objects that have been modified by a user, and they are updated each time the system is upgraded.

**Update versions table**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A unique identifier for coalescing versions of the same customized record.</td>
</tr>
<tr>
<td>Record name</td>
<td>Name of the customized record.</td>
</tr>
<tr>
<td>Source</td>
<td>Indicator of how the version was added on the instance.</td>
</tr>
<tr>
<td></td>
<td>- System Upgrade: from a software upgrade (the baseline version).</td>
</tr>
<tr>
<td></td>
<td>- Update Set: from an update set that was created or committed on the instance.</td>
</tr>
<tr>
<td></td>
<td>- Pull History: from a pull in Team Development.</td>
</tr>
<tr>
<td>State</td>
<td>Indicator of whether the version is or has ever been loaded on the instance.</td>
</tr>
<tr>
<td></td>
<td>- Current: the version is currently loaded.</td>
</tr>
<tr>
<td></td>
<td>- Previous: the version has previously been loaded on the instance.</td>
</tr>
<tr>
<td></td>
<td>When a current version is replaced by a new version, it becomes a previous version.</td>
</tr>
<tr>
<td></td>
<td>- History: the version was never loaded on the instance and was only inserted for historical purposes, such as when pulling versions from the parent in Team Development.</td>
</tr>
<tr>
<td>Application</td>
<td>The application for the customized record, if it is assigned to an application.</td>
</tr>
<tr>
<td>Payload</td>
<td>The data for this version of the customized record.</td>
</tr>
<tr>
<td>Additional fields on the list view</td>
<td>A reference to the older version record, if this version was created by reverting to an older version.</td>
</tr>
<tr>
<td>Instance Name</td>
<td>The name of the remote instance where the version was originally created.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Instance ID</td>
<td>The URL of the remote instance where the version was originally created.</td>
</tr>
<tr>
<td>Related lists on the form view</td>
<td>All versions of the customized record that are available on the instance.</td>
</tr>
</tbody>
</table>

**Merge tool**

The Diff Merge tool enables administrative users to compare differences between two versions of a record. Administrators can compare field-level changes between two versions, apply changes using Move Right field-level copy functionality and then merge results, or choose to revert to the non-current version. You can access the Diff Merge tool by comparing versions, resolving conflicts, or resolving collisions, during development or after upgrades.
<table>
<thead>
<tr>
<th>Action</th>
<th>INSERT OR UPDATE</th>
<th>INSERT OR UPDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessible from</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client callable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Description a</td>
<td>Description b</td>
</tr>
<tr>
<td>Name</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Script</td>
<td><code>var A = Class.create(); A.prototype = Object.extend(ObjectAbstractAjaxProcessor, {}</code></td>
<td><code>var B = Class.create(); B.prototype = Object.extend(ObjectAbstractAjaxProcessor, {}</code></td>
</tr>
<tr>
<td>Protection policy</td>
<td>Read-only</td>
<td>Read-only</td>
</tr>
<tr>
<td>Replace on upgrade</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Click to edit:**

```
var A = Class.create();
A.prototype = Object.extend(ObjectAbstractAjaxProcessor, {
```

```
var B = Class.create();
B.prototype = Object.extend(ObjectAbstractAjaxProcessor, {
```
Accessibility Functions

The platform includes accessibility features that support Web Content Accessibility Guidelines (WCAG) 2.0 level A and make the interface accessible to users with disabilities. These features improve the user experience when accessing platform functions with screen readers and keyboard navigation.

In general, you can use the following set of standard keyboard navigation functions:
• Press Tab to navigate major groupings in a pre-defined sequence, including moving between standard interface controls (fields and lists) in a module, or between records within a tab.

Press Shift Tab to move backwards in a pre-defined sequence.

Visually impaired users can navigate the Diff Merge tool. Screen readers can read all critical page content. All links and buttons can be reached when a section that is critical must be read. VoiceOver audible cues describe the content of the section that is necessary to read.

To enable accessibility functions, administrators should set these sys_properties:

1. Setting glide.ui.javascript_editor to false makes the following functions accessible:
   • Script fields (such as Script Include).
   • Side-by-side script comparison.

2. Setting sys_properties color settings enables high contrast visibility, which makes the left and right columns more accessible and easier to read by visually impaired users.
   • mergetool.bg.left.highlight - Left column cell color when values differ between versions.
   • mergetool.bg.right.highlight - Right column cell color when values differ between versions.
   • mergetool.bg.left - Left column cell color when version values are the same.
   • mergetool.bg.right - Right column cell color when version values are the same.

Compare to the current version

You can compare a version to the current version for any customizable object that a user has modified, such as a form layout or business rule. You can also compare the local and current pulled version of an object in Team Development. Administrators can suppress versions for specific tables.

To compare a version to the current version of an object:

1. Open the Compare to Current page using one of the following methods:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>From a Versions list</td>
<td>Right-click the version and select Compare to Current.</td>
</tr>
<tr>
<td>From the Update Versions form</td>
<td>Click the Compare to Current related link.</td>
</tr>
</tbody>
</table>
2. The Compare to Current page highlights the fields that differ. Review the differences. You have the following options:
## ServiceNow Kingston Now Platform Custom Business Applications

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>To resolve the differences by choosing the previous version</strong></td>
<td>• Team Development: Click <strong>Revert to Selected Version</strong>.</td>
</tr>
<tr>
<td></td>
<td>• For a version of an object: Click <strong>Use Local Version</strong> to maintain the local record as the current version. The pulled version is added to the version history for the record.</td>
</tr>
<tr>
<td><strong>To resolve the differences by modifying the current version and saving the merged changes</strong></td>
<td>You can either update the setting in the current record or move a setting from the selected version to the current version. To move a change, click the &gt; (Move Right) button for the field in the diff/merge tool. To work with scripts and text fields, click in the field and modify the text as needed. When the records meet your needs, click:</td>
</tr>
<tr>
<td></td>
<td>• Team Development: Click <strong>Save Merge</strong> to save the changes to the current version.</td>
</tr>
<tr>
<td></td>
<td>• Team Development: Click <strong>Use Pulled Version</strong> or <strong>Use Local Version</strong> option to accept or reject all changes, as appropriate.</td>
</tr>
<tr>
<td></td>
<td>• On the upgrade history Compare to Current form, the only option is <strong>Revert to Base System</strong></td>
</tr>
</tbody>
</table>

### Revert a change

You can undo changes to a customized record by reverting to an older version.

1. View a list of version records for an object.
2. Optional: Compare the current version to the older version to ensure that you are reverting the desired changes.
3. Right-click the older version and select **Revert to this version**. A confirmation dialog box appears.
   
   If reverting to this version results in data loss due to a database schema change, a warning message appears in the dialog box.
4. Click **OK** to confirm the action.
   
   • The current version is marked as a previous version.
   • A new version record is added that duplicates the version that you selected in the preceding step. This new version is marked as the current version.

   **Note:** You can revert to the most recent baseline version. You cannot revert to an older baseline version.

### Suppress versions

Administrators can configure a table so that it does not track customizations in the Versions [sys_update_version] table.

**Warning:** If you suppress versions for tables, Team Development may work incorrectly, and you may be unable to compare and revert versions of records on the tables.

1. Navigate to **sys_properties.list**.
2. Create a new property:
- Name: glide.update.suppress_update_version
- Type: string
- Value: a comma-separated list of tables. The default value is sys_user,sys_import_set_row.

**Versions and local changes**

Version records track changes to a customizable record over time so that you can compare or revert to a specific version later.

A version record is created every time a developer changes a customizable record, so a single record can have multiple version records associated with it. A local change record is created or updated to reference the current version every time a developer modifies a customizable record, so a single record can have only one local change record associated with it.

Local change records track which customized records have changes on the development instance that are not on the parent instance so that you can collect changes in preparation for a push.
Team Development concepts
Developers can back out a local change to restore a previous version of a customizable record. The back out action sets the local customizable record to the last revision identified by a reconciliation action.

**ServiceNow application repository**

After you develop and test a custom application, you can make the application available to company instances by publishing it to the ServiceNow application repository.

The ServiceNow application repository is a central repository for all scoped applications that are published by all ServiceNow customers. The application repository allows ServiceNow customers to upload and distribute applications between their instances. When you access the application repository, you can see and manage only the applications that are published by your own organization. You can’t see or manage applications that are published by other organizations.

After you have designed, developed, and successfully tested a custom application, you can publish your application to the ServiceNow application repository to share it to other instances in your company.

**Entitlements**

An entitlement refers to permission given to an instance to install a scoped application from the application repository. An instance must be entitled to an application in order for you to be able to install the application on the instance.

By default, after you publish an application to the application repository, all your company instances are entitled to the application automatically. To limit which company instances are entitled to the application, access the application repository by going to [https://apprepo.service-now.com](https://apprepo.service-now.com), and then change the entitlement type for the application. You can also entitle an instance again if the application entitlement has already been removed. For more information, see Manage application entitlements from the application repository.

**Using the application repository**

You can access the application repository by going to [https://apprepo.service-now.com](https://apprepo.service-now.com).

After you publish an application to the application repository, you can:

- Install an application from the application repository
- Manage application entitlements from the application repository
- Delete an application from the application repository
- Release a scope from the application repository
- View scopes that are available to your company
- View keys that are available to your company

**Publish an application to the application repository**

Publish a custom application to the application repository so that it can be installed on other instances in your organization.

To allow a developer to publish an application to the application repository, delegate the Publish to App Repo permission to the developer. For more information, see Add a developer.

Role required: admin, or delegated_developer with Publish To App Repo permission enabled

1. Navigate to **System Applications > My Company Applications**.
2. Open the In Development tab.
3. Open the application record that you want to publish to the application repository.
4. Click the Publish to My Application Repository related link.
5. Click Submit.

Install the application on company instances so that your organization can start using it. For more information, see Install an application from the application repository.

By default, after you publish an application to the application repository, all your company instances are entitled to the application automatically. To limit which company instances are entitled to the application, access the application repository by going to https://apprepo.servicenow.com, and then change the entitlement type for the application. For more information, see Manage application entitlements from the application repository.

Install an application from the application repository

Install your application on an instance so that employees can start using the application that you developed.

- Publish an application to the application repository.
- Check the entitlement type of the application to ensure that your instance is entitled to the application. For more information, see Manage application entitlements from the application repository.

Role required: admin

1. Navigate to System Applications > Applications.
2. Find the application.
3. Next to the application listing, select a version to install.
4. Click Install.

Manage application entitlements from the application repository

Add or remove application entitlements to limit which instances the application can be installed on.

You can manage only the applications that you’ve published to the application repository. For more information, see Publish an application to the application repository.

Role required: none

By default, after you publish an application to the application repository, all your company instances are entitled to the application automatically. To limit which company instances are entitled to the application, access the application repository by going to https://apprepo.servicenow.com, and then change the entitlement type for the application. You can also entitle an instance again if the application entitlement has already been removed.

2. Log in using your HI credentials.
3. Next to the application listing, click Select Action and then click Manage Entitlements.
4. Choose an entitlement type for your application.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove all existing entitlements</td>
<td>None of your company instances can install the application.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Entitle all instances</td>
<td>Any of your company instances can install the application. This option is selected by default when you publish an application to the application repository.</td>
</tr>
<tr>
<td>Entitle selected instances</td>
<td>Only instances that you select can install the application. Pick which instances to entitle by moving instances from the Available Instances list to the Selected Instances list.</td>
</tr>
</tbody>
</table>

5. Click OK.

Delete an application from the application repository

Delete an application from the application repository so that it’s no longer available to your company instances.

- Publish an application to the application repository
- You can delete an application only if the application is not installed on any of your company instances. Uninstall the application on all your instances before deleting it from the application repository.

Role required: customer_admin on HI

2. Log in using your HI credentials.
3. Next to the application listing, click Select Action and then click Flag for Deletion.
4. On the confirmation window, click Yes. After you flag an application for deletion, the application is deleted automatically after 90 days.
5. To delete the application immediately:
   a) Open the Flagged Apps tab.
   b) Next to the application listing, click Select Action and then click Delete Immediately.

Release a scope from the application repository

Release a scope from the application repository so that the scope can be used to create new scoped applications.

You can release an application scope only if the scope isn’t being used by any application. If your scope is being used by an application, follow the steps in Delete an application from the application repository before releasing the scope.

Role required: customer_admin on HI

The application repository stores the scopes of all your custom applications. You cannot create a new application using a scope that is being stored in the application repository. To release a scope means to remove the scope from the application repository so that you can create new applications using the scope.

2. Log in using your HI credentials.
3. Open the Scopes tab.
4. Next to the scope listing, click the trash icon (🗑).

**System update sets**

An *update set* is a group of configuration changes that can be moved from one instance to another. This feature allows administrators to group a series of changes into a named set and then move them as a unit to other systems for testing or deployment.

An update set is an XML file that contains:

- A set of record details that uniquely identify the update set.
- A list of configuration changes.
- A state that determines whether another instance can retrieve and apply configuration changes.

Update sets track changes to applications and system platform features. This allows developers to create new functionality on a non-production instance and promote the changes to another instance.

**Note:** Properties that are tagged as *Private* are excluded from update sets. Keeping system properties private prevents settings in one instance from overwriting values in another instance. For example, you may not want a system property in a production instance to use a particular value from a development instance.

Application developers have additional options with update sets such as:

- Create an update set for a specific version of an application.
- Specify which application tables to track in update sets.

**Update set tables**

Each update set is stored in the Update Set [sys_update_set] table, and the customizations that are associated with the update set, which are entries in the Customer Update [sys_update_xml] table, appear as a related list on the update set record.

When a tracked object is customized, a corresponding record is added or updated in the Customer Update [sys_update_xml] table and is associated with the user current update set. The *associated application file properties* are tracked and transferred along with the customized object in a single update record. A corresponding record is also added to the Versions [sys_update_version] table.

The Customer Update table contains one record per customized object, per update set. The Versions table contains one record per change to a customized object.

- Administrators can compare two versions and revert to a specific version of an object.
- Administrators can suppress versions for specific tables.
- Administrators can specify fields on tracked tables that you can change without skipping updates to the rest of the record (exclude the field from the update).

**Note:** Do not directly modify Customer Updates [sys_update_xml] records.
Customizations tracked by update sets

Update sets can track customizations to application tables, fields, and records.

Update sets track customizations under these conditions:

- Where the table has an `update_synch` dictionary attribute.
- Where there is a special handler to track changes to multiple tables.
- Where the administrator has not excluded a field from updates.

In general, update sets capture configuration information but not task or process data. For example, update sets track service catalog item definitions and related configuration data like variables and variable choices. However, if you test the service catalog by placing orders, update sets do not track order requests, items, and catalog tasks.

Update sets have a limited capacity to transfer data as application files. For larger data transfers, export data and import it with an import set or web service.

update_synch attribute

To view the list of tables where customizations are tracked, navigate to System Definition > Dictionary and filter on `attributes CONTAINS update_synch`.

⚠️ Warning: Do not add the `update_synch` attribute to a dictionary record. When improperly used, this attribute can cause major performance issues or cause the instance to become unavailable. Adding this attribute is not supported.

A default rule blocks the use of the `update_synch` attribute on a table for which it is not predefined to avoid the following issues:

- Some core tables require special update handling because they represent information on multiple tables. When the `update_synch` attribute is added to these tables, duplicate update records are created, causing major conflicts that are difficult to troubleshoot and repair.
- Using the `update_synch` attribute to migrate data records between instances can cause performance issues, because it is not intended for this purpose. To migrate data, use an instance-to-instance import.
  
  See Import Sets key concepts.

Special handlers

Some changes require special handlers because they represent information on multiple tables. These changes are packaged into one update set entry so that all records are properly updated when the customization is committed. The following changes are tracked with special handlers:

- Workflows
- Form sections
- Lists
- Related lists
- Choice lists
- System dictionary entries
- Field labels

Choice lists
Update sets store both new and updated choice options as separate records in the Update Version [sys_update_version] and Customer Update [sys_update_xml] tables. For example, you create a new Activity [u_activity] table that extends the Task table. You then add a new choice option to the Task state field that is only visible for your extended table (for example, My State).

When you publish these changes as an update set, the update only contains update and version records for the choice you added to the u_activity table. The choice options in the task table are unaffected.

Dictionary changes

Usually, using update sets prevent you from applying dictionary changes that result in data loss. Blocked dictionary changes include:

- Removing tables
- Changing a column data type

Update sets do not track the removal of tables from the system dictionary. Instead, customers must manually remove tables from the target instance.

While update sets track data type changes, the target instance skips any change that results in data loss and instead adds a log message about the action. Customers can use the log to manually make data type changes on the target instance.

Note: Update set previews do not check for type mismatch problems since the target instance skips changes resulting in data loss. Also, using update sets to delete a column from a table can cause data loss in certain circumstances. If there is data in the column on the target instance, that data is deleted, as well as the column itself, when the update set is committed. A warning message appears if you attempt to commit an update set that deletes a column. The message states that there are one or more delete updates that cause the data to be deleted, and it specifies what delete updates there are.

Homepages and content pages

Homepages and content pages are not added to update sets by default. Add pages to the current update set by unloading them.

Application changes

The system creates a separate update set for each application that only contains changes associated with the application. This ensures that access settings for each application are properly evaluated and applied when committing update set changes.

Default update set

Only one update set can be the default set for any application scope.

To set an update set to be the default set, you set the Default set field to true. When you set Default set = true, the following actions occur:

- The update set becomes the default update set for its scope.
- The system sets Default set = false for all other update sets with the same scope. This ensures that there is only one default update set for each scope.
Global default set

Use the global default update set to make changes to an instance without adding the changes to any user-created update sets. The global default update set is the set where **Default set = true** and application scope is **global**. The global default set (regardless of the **Name** of the set) provides system functionality and should not be changed, deleted, or moved between systems. Use this update set to make changes to an instance without adding the changes to any user-created update sets.

Auto-generated default set

At all times, to ensure that no updates to an instance are lost, the system ensures that there is a default set for the user’s current scope. If the system finds that a default update set does not exist (or is marked **Ignored** or **Completed**) for the current scope, then the system auto-generates an update set and sets **Default set = true**.

These are some common cases where the system auto-generates a default update set.

- The very first time that an admin logs in, the system sets the system’s global default update set as the administrator’s update set. In addition, the application picker sets the administrator’s application scope to global.

  If a global default update set does not exist (or is marked **Ignored** or **Completed**), the system creates a new update set for the global application scope and performs the following actions:

  - The system sets **Default set = true** for the new set.
  - The system sets the name of the new set to start with the name of the former default set and appends the next numeral (in the sequence SetName, SetName 1, SetName 2, ..., SetName n).
  - The system sets the newly created set as the administrator’s update set.

- When a user marks the default set for a scope as **Ignored** or **Completed** (not a recommended practice), the system immediately auto-generates a new default set for the scope.

- The system auto-generates a new default update set for a scope when all the following conditions occur:

  - You change application scope.
  - Your preferred update set is **Complete** or **Ignored**.
  - There is no In-Progress default update set for the new scope.

Get started with update sets

Because update sets make changes to an instance, review this information to avoid errors and performance issues. Learn how to plan the update process and avoid common mistakes.
## When to use update sets

<table>
<thead>
<tr>
<th>Deployment option</th>
<th>Good for</th>
<th>Future considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update Sets</td>
<td>Storing changes to a baseline or installed application.</td>
<td>You can manually create update sets to store a particular application version.</td>
</tr>
<tr>
<td></td>
<td>Storing and applying a particular version of an application.</td>
<td>Use update sets to deploy patches or changes to installed applications.</td>
</tr>
<tr>
<td></td>
<td>Producing a file for export.</td>
<td><strong>Note:</strong> Do not use update sets to install applications. Instead, use the application repository or the ServiceNow Store to install applications.</td>
</tr>
<tr>
<td>Application Repository</td>
<td>Installing and updating applications on all company instances.</td>
<td>Consider uploading an application to the ServiceNow Store to share it with other users.</td>
</tr>
<tr>
<td></td>
<td>Automatically managing application update sets.</td>
<td>Allows installation of and update to the latest application version only.</td>
</tr>
<tr>
<td></td>
<td>Restricting access to applications to the same company.</td>
<td>Use update sets to store prior application versions.</td>
</tr>
<tr>
<td></td>
<td>Deploying completed applications to end users.</td>
<td><strong>Note:</strong> If used with team development, publish applications only from a parent instance.</td>
</tr>
<tr>
<td>Team Development</td>
<td>Providing change management across multiple instances.</td>
<td>Consider providing each development team access to a dedicated development instance.</td>
</tr>
<tr>
<td></td>
<td>Allowing multiple developers to work on applications.</td>
<td>Requires developers to manually merge colliding changes.</td>
</tr>
<tr>
<td></td>
<td>Organizations that have access to several non-production instances.</td>
<td>Works only for instances owned by the same organization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If used with the application repository, publish applications from a parent instance.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Plan the update process

Before working with update sets, create a standard process for moving customizations from instance to instance using this check list:

1. Check that both instances are on the same version. Customizations may not work if they rely on code that has changed between versions.

2. Determine the changes to make in a single update set. Complete your update sets as you finish small to medium-sized tasks. As update sets get larger, it becomes harder to review them, takes longer to identify specific changes within them, increases the risk of conflicts with other update sets, and takes more time to preview and commit them. This is especially true if the update sets contain schema changes or revisions to large workflows or if the set has to be backed out.

3. Ensure that all base system records have matching sys_id fields. Some base system records are created on an instance after provisioning and do not match between different instances, leading to problems with update sets. The best way to avoid this issue is to:
   - Provision production and non-production instances.
   - Clone the production instance onto the non-production instance.

4. Identify a common path for update sets to move from instance to instance and maintain that model. Never migrate the same update set from multiple sources. Move update sets from dev to test and then from test to production.

5. Plan for when to commit the update sets to production. Avoid committing an update sets to a production instance during business hours. The instance may perform slower temporarily as the update sets applies.

6. Make sure update set names are clear. Create a naming convention to coordinate changes from multiple developers and to reference when committing the changes to another instance.
   - If update sets are being generated as fixes for problems, consider including the problem ticket in the name (for example, PR10005 - Duplicate Email Issues Fix).
   - If you need more than one update set to address a problem, include a sequence number in the naming convention so that update sets are applied in the order that they were created (for example, PR10005 - Duplicate Email Issues Fix and PR10005.2 - Duplicate Email Issues Fix).

7. Understand the following about update sets:
   - What records are generated.
   - Which customizations are tracked.
   - Which dictionary changes are valid.
   - Which customizations can be backed out (reversed) once applied.

8. Before making any customizations, double-check that the correct update set is selected.

Working with update sets

Review this information to avoid errors and performance issues.

- Do not delete update sets. If an update set is deleted, any updated records may be overwritten in the next update.
- Do not include the `system_id` field from the `ldap_server_config` record in an update set. An update set from a working configuration points to the wrong `system_id` node for the target instance and does not work.
- Do not back out the Default update set. This action causes damage to the system.
- Never change the `Update Set` field value (update_set) in a Customer Update record (`sys_update_xml`). If a customization is made in the wrong update set, take the following action:
  1. Switch to the desired update set.
  2. Modify the object (record) that was originally changed. You can make a trivial change, such as adding a field.
  3. Save the record.
  4. Back out the change just performed, and then save the record again.

  This action ensures that the latest version of the object is included in the desired update set and prevents duplicate updates for the same object in a single update set.

- Do not mark an update set as Complete until it is ready to migrate. Once an update set is complete, do not change it back to In progress. Instead, create another update set for the rest of the changes, and make sure to commit them together in the order that they were created. Naming conventions may help in this case (for example, Performance Enhancements and Performance Enhancements 2).
- Do not manually merge updates into an update set. Always use the Merge Update Sets module. This tool compares duplicate files between update set and selects the newest version.
- If a committed update set has a problem in the test instance, build the fix in another update set in the development instance. Commit this set to the test instance, and then make sure both sets are migrated to the production instance and committed in the order they were made.
- Always preview an update set before committing it.
- Set completed update set on the production instance to Ignore. This state ensures the update set is not reapplied when cloning the instance.
- Keep a to-do list of manual changes and data loads that need to be completed after an update set is applied.
- Do not make too many changes at one time. Verify that the correct changes have been made incrementally.

**Update set administration**

Administrators can configure options for update sets, such as excluding certain fields from updates and controlling access to the update set picker.

Administrators have the following options with update sets.

- Create an update set to store local changes.
- Select the current update set to store local changes.
- Commit an update set to prepare it for distribution.
- Compare update sets to determine what differences they contains.
- Merge separate update sets into a single update set.
- Create an external file from an update set.
- Retrieve update sets from remote instances.
- Apply retrieved update sets.
- Back out changes applied from an update set.
- Set system properties related to update sets.
Exclude a field from an update

Administrators can specify fields on tracked tables that you can change without skipping updates to the rest of the record. To exclude a field from updates, add the `update_exempt` dictionary attribute to the field.

During subsequent software upgrades, the value of the excluded field is preserved, while the rest of the record receives updates. For example, you may want to select the `Client callable` check box for a script include, but still receive upgrades to the script.

Note: Values for excluded fields are not retained when you revert customizations to a default software version. For example, you activate a UI macro and change the XML script. Later, a software upgrade contains a feature for the macro that you would like to implement, so you revert your customizations. The default version replaces the entire customized version, and you now need to reactivate the macro. If you also change a field that is not update_exempt, then updates are skipped for the entire record (the entire customization is preserved during upgrades).

Track the Active field

The system automatically treats the `Active` field on a tracked table as update_exempt even if the attribute is not present. This allows you to change the field value without affecting the `Updated` and `Updated By` system fields.

To specify that changing the `Active` field preserves the entire record (it is not excluded), add the following attribute to the `Active` field on the table: `update_exempt=false`.

Track an application table

Application developers can track application changes in an update set to save or distribute a particular version of an application. During table creation, set `Extends Table` to Application File (sys_metadata).

Grant access to the update set picker

The update set picker allows users to choose an update set for making and tracking customizations. By default, only administrators can use the update set picker. You can grant access to additional users.

Automatically preview retrieved update sets

By default, the system automatically previews update sets as soon as it has retrieved them. To turn off this behavior, set the system property `glide.update_set.auto_preview` to `false`. In the navigator filter, type `sys_properties.list` then navigate to the `glide.update_set.auto_preview` property and edit the value field.

Grant access to the update set picker

Enable a non-administrative user to use the update set picker.

Role required: admin
The update set picker appears on the Settings panel. The picker allows users to choose an update set for making and tracking customizations. By default, only administrators can use the update set picker. You can grant access to additional users.
1. **Grant the user role read access** to the Update Set table (sys_update_set).
2. Enable users to see the update set picker on the Settings panel.
   a) **Add the system property** `glide.ui.update_set_picker.role` **to the System Properties table.**
b) Set the value of `glide.ui.update_set_picker.role` to the role for which you want to give access.

**Overwrite customizations during an upgrade**

When you change any non-excluded fields on a record, a corresponding record is added in the Customer Update `[sys_update_xml]` table and the `Replace on upgrade` field is set to `false`.

To prevent customizations from being overwritten by system upgrades, the upgrade process automatically skips changes to these objects. You may want to overwrite your customizations with the next software version. For example, you may change a script to implement a temporary workaround for a problem that is fixed in the next version. You would want to overwrite your workaround when upgrading to the next version to ensure that you receive any future enhancements to the script.

1. Open the customized object (for example, the `ArrayUtil` script include).
2. Right-click the header and select `Show Latest Update`.
3. Configure the form to add the `Replace on upgrade` field, if necessary.
4. Select the `Replace on upgrade` check box and click `Update`.
   The customized object will be replaced on the next upgrade.

**Creating, testing, and moving customizations**

Use these procedures to create, test, and move customizations from a development system to a production system.

**Three-step import process**

A common process for developing customizations with update sets involves moving changes from development, to test, to production instances.

1. Create an update set on the development instance.
2. Make customizations and changes on the development instance.
3. Mark the update set as Complete.
4. Log in to the test instance and retrieve the completed update set from the development instance.
5. Commit the update set on the test instance, and test customizations thoroughly.
6. If the update set has problems in the test instance, repeat steps 1 - 5 to develop the fix on the development instance with another update set.
7. Log in to the production instance and retrieve the completed update set from the development instance. If the update set required a fix, retrieve both update sets.
8. Commit the update set on production. If the update set required a fix, commit both update sets in the order they were made.

**Two-step import process**

If your development environment consists of only two instances, you can combine your development and testing instances into a single staging instance.

1. Create an update set on the staging instance.
2. Make customizations and changes on the staging instance.
3. Mark the update set as Complete.
4. Test customizations thoroughly on the staging instance.
5. If the update set has problems, repeat steps 1 - 4 to develop the fix on the staging instance with another update set.
6. Log in to the production instance and retrieve the completed update set from the staging instance. If the update set required a fix, retrieve both update sets.
7. Commit the update set on production. If the update set required a fix, commit both update sets in the order they were made.

**Update set use**

These procedures help you manage your customizations and resolve potential collisions before you move them to another instance.

Before using update sets, review *Get started with update sets* to learn when to use and when not to use update sets, and how to plan the update process and avoid common mistakes. Then, create an update set and use it to make changes on a development instance. You can report on updates, merge update sets, and compare update sets to ensure the desired changes are ready to move.

When the update set is completed, you can transfer the update set to another instance according to your test process. See *Update set transfers* for details.

**Create and select an update set as the current set**

Create the update set you need for your customizations and select it as the current set.

You can configure an update set as the current set when you create it or select it later from the Settings panel.

1. Navigate to System Update Sets > Local Update Sets and click New.
2. Complete the form from the fields in the table.
3. Click Submit to create the update set. If the picker is enabled and the update set is in the In progress state, click Submit and Make Current to:
New update set

Update Set Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a unique name for the update set. You can use naming conventions to organize update sets. For example, add the problem number to the name of the update that fixes it, identify the application scope, or use sequence numbers to keep track of the order in which update sets need to be committed.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>State</td>
<td>Select <strong>In progress</strong> for a new update set. Selecting an <strong>In progress</strong> update set tracks customizations in the update set. The <strong>update set picker</strong> only displays <strong>In progress</strong> update sets. Select <strong>Completed</strong> only when you are certain that the update set is complete. After it is marked <strong>Completed</strong>, do not set it back to <strong>In progress</strong>. Instead, create a new update set with further customizations, and make sure to commit the update sets in the order that they were marked <strong>Completed</strong>. Use <strong>Completed</strong> update sets to transfer changes from one instance to another. Select <strong>Ignore</strong> when you are no longer working on an update set but do not want it to be transferred to another instance. You should always set <strong>Completed</strong> update sets on the production instance to <strong>Ignore</strong>. This state ensures the update set is not committed again when cloning the instance. Update set picker only displays <strong>In progress</strong> update sets. See <strong>Select the current update set in System Settings</strong>. Select <strong>Completed</strong> only when you are certain that the update set is complete. After it is marked <strong>Completed</strong>, do not set it back to <strong>In progress</strong>. Instead, create a new update set with further customizations, and make sure to commit the update sets in the order that they were marked <strong>Completed</strong>. Use <strong>Completed</strong> update sets to transfer changes from one instance to another. Select <strong>Ignore</strong> when you are no longer working on an update set but do not want it to be transferred to another instance. You should always set <strong>Completed</strong> update sets on the production instance to <strong>Ignore</strong>. This state ensures the update set is not committed again when cloning the instance.</td>
</tr>
<tr>
<td></td>
<td><strong>Release date</strong> Enter the date on which you plan to release the update set. <strong>Application</strong> Populates the application scope that is currently selected in the Application picker. All changes in the update set apply only to the current scope. <strong>Description</strong> Enter a description of the update set.</td>
</tr>
</tbody>
</table>

**Select the current update set in System Settings**

You can change the current update set at any time, using the update set picker in the System Settings panel.
Role required: admin

1. To select the update set from the picker, click the gear icon in the header bar to open the System Settings panel.

2. In the System Settings panel, select Developer.

3. Select the update set you want to make current from the choice list in the Update Set field and close the settings panel.
Update set picker
View customizations and compare with current version

View the customizations that make up an update set and compare the update to the current version.

Role required: admin

The Customer Update (sys_update_xml) table contains one record per customized object. The customer update record specifies:

- The update set containing the customized object.
- The type of action applied to the customized object.
  - INSERT
  - INSERT_OR_UPDATE
  - UPDATE
  - DELETE
- The type of object customized.
- The target object of the update.
- The sys_id of the customized object (if it is a change to a particular record).
- The user who customized the object.
- The date and time the object was customized.

1. Navigate to System Update Sets > Local Update Sets.
2. Click the update set name.
3. View the Customer Updates related list.
   
   You can compare any update to the current version. Right-click the update record and select Compare to Current.
### Incident Release 1

**Name:** incident Release 1

**State:** In progress

**Application:** Global

**Created:** 2017-10-10 09:27:02

**Created by:** admin

**Merged to:**

**Description:** Tailoring incident for a new group that is coming on board.

### Related Links

- Make This My Current Set
- Merge With Another Update Set

### Customer Updates

<table>
<thead>
<tr>
<th>Customer Updates</th>
<th>Update Set Logs</th>
<th>Child Update Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Search</td>
<td></td>
</tr>
</tbody>
</table>

### Updated Set: Incident Release 1

- **Created:** about a minute ago
  - **Type:** Table
  - **View:** Incident Table 1
  - **Target name:** admin

- **Created:** about a minute ago
  - **Type:** Dictionary
  - **View:** Incident Table 1
  - **Target name:** admin
Navigation between records

You can navigate between a customer update record and the customized object or the application file for the object.

Navigate from an update record to:

- The customized object, such as the application menu record: Click the Show Related Record related link.
- The application file record for the object: Click the Show Application File related link.

Navigating to the application file

To navigate from a customized object or an application file to the current customer update record: Click the form header and selecting Show latest Update.
Navigating to the customer update record

View a report on customizations and configuration changes

The base system provides reports for changes to the Incident table and changes by the current user.

1. Navigate to Reports > View / Run and locate the Customer Update section.
2. Run any of the available reports or create a new report. The following reports are available:
   - **Application Changes (Incident)**: Displays all changes made to the Incident table. Select a different table and run the report again to view all changes to another application.
   - **My Changes**: Displays all changes created or updated by the current user, grouped by table name.

Merge update sets

You can merge multiple update sets into a single update set. This capability is supported for backward compatibility with earlier releases of the ServiceNow® platform. The newer batch update sets feature accomplishes the same outcome with a more predictable and robust solution.

**Note:** You cannot “unmerge” update sets once they have been merged.

1. Navigate to System Update Sets > Merge Update Sets. By default, the list is filtered to only show update sets that are In progress.
   Alternatively, navigate to System Update Sets > Merge Completed Sets. By default, the list is filtered to only show update sets that are in the Complete state. For example, you may want to use this filter after pushing changes or transferring update set from a development to a test instance.
2. Filter the list to show only the update set that you want to merge. You can only merge update sets that belong to the same application.
### Merge update sets

3. Enter a **Name** for the new update set. Updates are moved to this new update set during the merge process.

4. Optional: Enter a **Description** for the update set.

5. Click **Merge**.
6. In the confirmation dialog box, click **OK**.
   - The new update set is created.
   - The most recent change for each object is moved from the original sets to the new set. Only changes that are not merged into the new set remain in the original sets. A message indicates how many updates were moved and how many were skipped. For example, if both update sets modify the Incident form, only the most recent change is moved to the new update set. The other modification remains in its original update set to provide a record of the changes that were not moved.

   **Note:** The system determines which record is the most recent by comparing the **Updated** field for the records, NOT the **sys_updated_on** value in the payload.

7. (Recommended) Verify that the correct changes were moved to the new set by scrolling down to the **Merged Update Sets** related list and opening the old update set records.

8. (Recommended) Delete or empty the original update sets to avoid committing an older change by mistake. The system does not remove updates that were not merged into the new set. **DO NOT** move updates "left behind" in old sets into the new set.

**Compare local update sets**

Administrators can preview local and remote (retrieved) update sets and compare the sets with one another to resolve conflicting changes.

Compare local update sets to identify collisions and ensure that the proper changes are being committed. Resolve all conflicts before moving an update set between instances.

1. Navigate to **System Update Sets > Local Update Sets**.
2. Select the check boxes beside the update sets to compare.
3. In the **Action** choice list, select **Compare Update Sets**.

The progress screen appears as ServiceNow generates the collision report.

<table>
<thead>
<tr>
<th>Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Generating Collision Report</td>
</tr>
<tr>
<td>State: Complete</td>
</tr>
<tr>
<td>Completion code: Success</td>
</tr>
<tr>
<td>Message: Collision Report COL0001002 created</td>
</tr>
</tbody>
</table>

**Collision report**

4. Click **Go to the Collision Report** when the report is complete.

   The Update Set Collisions list appears, showing all the changes in the selected sets.

5. Inspect the list for collisions by locating duplicate **Collision Numbers** that show the same change in separate update sets.
Update set collisions

6. Resolve the collision by deleting the unwanted update record from one of the update sets.
   1. Click the link in the Sys update column for the unwanted update (sys_ui_list_incident_null in the example).
   2. Click Delete.

   Note: You must open the update record to delete the record. You cannot delete the update by selecting the check box for the entry in the Update Set Collisions list and using the Delete action. When you delete the update record, the customization is not backed out of the instance. Only the record of the customization is deleted.
7. Run the comparison again to make sure all collisions have been resolved.

**Update set collision resolution**

A collision is an update that has a newer local update.

The platform detects collisions by comparing the values in the **Name** and **Updated** fields of the customer update record from each update set. If the name matches but there are different update date values, then there is a collision.

When a customer update is moved from one instance to another, it may be re-written to match the target instance. The re-write can involve changing the update name of the customer update and one or more **sys_ids** within the update. The re-writes are done when the record or the reference field is for a table that uses a coalesce strategy. This ensures that the customer
update will be applied to the correct record. For example, if the sys_dictionary record for tablename.fieldname has sys_id 123456789 on instance A and sys_id 987654321 on instance B, when a customer update that refers to that record is retrieved from instance A and recorded in the sys_update_xml table on instance B, references to 123456789 are updated to read 987654321.

Coalesce strategies

Update sets can detect collisions between identical records that you independently create on separate instances. To detect such collisions, the record must have a coalesce strategy based on coalescing columns. Because collision detection depends on uniqueness of tables, the tables must be unique when the coalescing columns are combined. Records that are not listed here will not collide if the same record is created separately on different instances.

<table>
<thead>
<tr>
<th>Type</th>
<th>Coalescing Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_db_object</td>
<td>name</td>
</tr>
<tr>
<td>sys_dictionary</td>
<td>name, element</td>
</tr>
<tr>
<td>sys_choice_set</td>
<td>name, element, language</td>
</tr>
<tr>
<td>sys_documentation</td>
<td>name, element, language</td>
</tr>
<tr>
<td>sys_properties</td>
<td>name</td>
</tr>
<tr>
<td>sys_report_chart_color</td>
<td>name, element, value</td>
</tr>
<tr>
<td>sys_ui_form</td>
<td>name, view, sys_domain</td>
</tr>
<tr>
<td>sys_ui_message</td>
<td>documentkey, language</td>
</tr>
<tr>
<td>sys_ui_list</td>
<td>name, view, sys_domain, element, relationship, parent</td>
</tr>
<tr>
<td>sys_ui_section</td>
<td>name, view, caption, sys_domain</td>
</tr>
<tr>
<td>sys_ui_related_list</td>
<td>name, view, related_list, sys_domain</td>
</tr>
<tr>
<td>sys_ui_view</td>
<td>name</td>
</tr>
<tr>
<td>sys_user_role</td>
<td>name</td>
</tr>
<tr>
<td>sys_user_group</td>
<td>name</td>
</tr>
<tr>
<td>sys_wizard</td>
<td>name</td>
</tr>
</tbody>
</table>

How customer update record names affect collisions

To understand coalescing, it helps to understand how records that do not coalesce work. For most record types, when a customer update is moved to a new instance, the system does not detect collisions for the following reason:

- When you create a record, it receives a unique sys_id. For most record types, the sys_id becomes part of the customer update record name. For example: sysevent_email_template_9e1998c078b71100a92ecacd80df1d39.
- Creating an identical record in the same table on another instance produces a customer update record name with a different sys_id. For example: sysevent_email_template_10b958c8653311005840134572f8e020
As a result, even though the records might be otherwise identical, the records have different names so the system does not detect the collision.

Coalescing records, in contrast, use the following approach to naming records and determining collisions: The following customer update record types use some or all of their coalescing columns instead of the sys_id in their names.

- sys_dictionary
- sys_documentation
- sys_choice_set
- sys_ui_list
- sys_ui_related_list

The resulting identical record name in each instance helps the system to identify collisions even if the records have different sys_ids.

When a customer update is moved from one instance to another, it may be re-written to match the target instance. The re-write can involve changing the update name of the customer update and one or more sys_ids within the update. The re-writes are done when the record or the reference field is for a table that uses a coalesce strategy. This ensures that the customer update will be applied to the correct record. For example, if the sys_dictionary record for tablename.fieldname has sys_id “123456789” on instance A and sys_id “987654321” on instance B, when a customer update that refers to that record is retrieved from instance A and recorded in the sys_update_xml table on instance B, references to “123456789” are updated to read “987654321”.

Preventing duplicate records

- Transfer data with update sets rather than recreating it on separate instances to ensure the records have the same sys_id.
- Export and import records as XML files to ensure the records have the same sys_id. See Export and import XML files.
- Enable a unique index for the table from the system dictionary. See Table administration.

Note: The default records included in the baseline system will always have the same Sys ID because the instance imports the records as XML files during instance provisioning.

Mark an update set complete

When you have completed the customizations and compared local update sets to resolve conflicts, mark the update set as Complete.

Mark an update set as Complete only when it is ready to migrate. Once an update set is complete, do not change it back to In progress. Instead, create another update set for the rest of the changes, and be sure to commit them together in the order that they were created. Naming conventions may help in this case (for example, Performance Enhancements and Performance Enhancements 2).

1. Open the update set record.
2. Change the State of the update set from In progress to Complete.
   - The update set is available for other instances to retrieve.
   - No additional customizations are tracked in the update set.
Save an update set as a local XML file

Administrators can export an update set as an XML file to save a specific version of an application or set of changes.

Typically you create an XML file of an update set when one of the following conditions apply:

- The two instances do not have network connectivity so you cannot retrieve update sets from the remote instance nor create a data source to pull, or import, data directly from the source instance.
- You do not want to provide administrator credentials to the source instance (for example, you do not want to share an administrator password with people outside your company) so you cannot retrieve update sets nor create a data source.
- You want to back up important customizations locally.

The ability to export and import customizations as an XML file is provided by the following UI Actions:

- Export to XML on the Update Set (sys_update_set) table.
- Export to XML on the Retrieved Update Set (sys_remote_update_set) table.
- Import Update Set from XML on the Retrieved Update Set (sys_remote_update_set) table.

The Export to XML UI Action on Update Set (sys_update_set) table calls a processor called UnLoadRetrievedUpdateSet, which transforms a local update set into a retrieved update set, exports the retrieved update set with its related list, and then deletes the temporary update set, if necessary.

Both Export to XML UI actions depend on the script include ExportWithRelatedLists, which exports a record and manually defined related lists to a single XML file.

1. Navigate to System Update Sets and click either Local Update Sets or Retrieved Update Sets.
2. Select an update set that is in the Complete state.
3. On the Update Set form, click the Export to XML Related Link.
4. Save the XML file.

An XML file is created. When the file is uploaded to another instance, it appears as a retrieved update set regardless of whether it is local or retrieved on the instance where it is created.

Load customizations from a single XML file

Administrators can load an update set XML file to apply a specific version of an application or set of changes.

1. Elevate privileges to the security_admin role.
3. Click the link Import Update Set from XML.
4. Click Choose File and select an XML file.
5. Click Upload.
The customization is now available as a retrieved update set with state **Loaded**.

6. Follow standard procedure to commit the update set.

![Screenshot of Retrieved Update Sets](image)

### Update set transfers

When an update set is completed, you can transfer it between instances to move customizations from development, through testing, and into production.

**Note:** Properties that are tagged as **Private** are excluded from update sets. Keeping system properties private prevents settings in one instance from overwriting values in another instance. For example, you may not want a system property in a production instance to use a particular value from a development instance.

#### Transferring with IP access control

If IP address access control is enabled on the source instance or the source instance resides in a different datacenter than the target instance, complete these tasks before transferring an update set:

1. Contact ServiceNow Technical Support to find out the IP addresses of all application nodes supporting your instance.
2. On the source instance, navigate to System Security/IP Address Access Control. Add the IP address from step one as an exception.

#### Transferring with basic authentication

If the source instance has basic authentication turned on for SOAP requests, you must use valid credentials to retrieve update sets.
Transferring with an XML file

You can unload an update set as an XML file and then transfer it to another instance. See Save an update set as a local XML file for details.

Retrieve an update set

Retrieve completed update sets from another instance.

1. If IP address access control is enabled on the source instance, set up the target instance as an exception.
2. On the target instance, navigate to System Update Sets > Update Sources and click New.
3. Specify the connection settings as described in the table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a unique name for the instance.</td>
</tr>
<tr>
<td>Type</td>
<td>Specify whether the remote instance is a development, test, or UAT instance.</td>
</tr>
<tr>
<td>Active</td>
<td>Specify whether the local instance can transfer update sets to the remote instance. You can transfer update sets only to active remote instances.</td>
</tr>
<tr>
<td>URL</td>
<td>Specify the URL of the remote instance using the appropriate transfer protocol. Each remote instance record should have a unique URL. Creating duplicate records with the same URL can cause errors. The remote instance must be on the same release family as the local instance.</td>
</tr>
<tr>
<td>Username</td>
<td>Enter the user on the remote instance who authorizes transferring update sets to this the instance. This user account must have the admin user role on the remote instance.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the password of the authorizing user.</td>
</tr>
<tr>
<td>Short description</td>
<td>(Optional) Enter any other relevant information about the remote instance.</td>
</tr>
</tbody>
</table>

4. Click Test Connection.
   - If the connection is successful, a confirmation message appears.
   - If the connection fails, a warning message identifies the cause of the failure.

5. If the connection fails, modify the settings to establish connectivity.
   - You must establish connectivity before you can save the connection settings.
   - You may want to modify the source instance (for example, change the password).
6. Right-click the form header and select Save.
   - Any update sets marked as Completed are transferred from the source instance to the target instance. Update sets that already exist on the target instance are skipped.
   - The confirmation page provides detailed messages about how many update sets were transferred and how many were skipped.
   - To view retrieved update set, navigate to System Update Sets > Retrieved Update Sets.

If the system property glide.update_set.auto_preview is set to true, the system automatically starts the preview process after the update set is retrieved. If this property is false, you must start the process manually. For more information on the preview process, see Preview a remote update set.

**Preview a remote update set**

Previewing compares an update set retrieved from a remote instance to updates on the local instance to detect potential problems. You must preview an update set and address all problems before you can commit the update set.

1. If the system property glide.update_set.auto_preview is set to true, the system automatically starts the preview process after the update set is retrieved. If this property is false, the preview process must be started manually.
   a) Navigate to System Update Sets > Retrieved Update Sets.
   b) Click Preview Update Set.

For large update sets, the preview process may require a significant amount of time. If necessary, you can cancel the preview process by clicking the Cancel button on the progress dialog box.

The Update Set Preview page shows results and lists problems. Read the information and click Close.

2. On the Retrieved Update Set form:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>If no problems were detected</td>
<td>Click Commit Update Set to commit all changes on the instance without reviewing the preview results.</td>
</tr>
<tr>
<td>If problems were detected</td>
<td>Address each problem in the Update Set Preview Problems related list.</td>
</tr>
</tbody>
</table>
To commit this update set you must address all related problems by fixing and previewing again. You may also just accept or skip the remote update that caused the problem.

Problems addressed: 0 Ignored | 0 Skipped | 5 Remaining

Name: Vehicle Update Set
Application: Global
Update source: Source
State: Previewed
Loaded: 2015-02-03 13:11:38

Description:

Application name: Global

Related Links
Show All Preview Records

Update Set Preview Problems (5)

Available Actions:
Find missing field
Find missing update
Accept remote update
Skip remote update

Could not find a table field (x_snc_test_vehicle_status) referenced in this update
3. Optional: Open the update set record and click **Show All Preview Records** to make sure the correct updates are being committed.

4. Optional: Open the update set record and click **Run Preview Again** to run the comparisons again. Review the **Update Set Preview Problems** related list to ensure that the correct updates are being committed.

**Review a preview record for an update set**
The process of previewing an update set creates a preview record for each update. You can review the preview records to make sure that the correct updates are being committed.

1. Open the Update Set record and preview the update set.
2. Click the **Show All Preview Records** related link.
3. Click the **Disposition** to open a preview record and then review the information (see table).
4. Fill in the fields on the form, as appropriate.
## Update Set Preview - Created 2015-02-04 13:44:37

### Disposition
- Collision/Update

### Proposed action
- Commit

### Remote update set
- Vehicle Update Set

### Local update

### Update set

#### Current Version
- 1: number
- 2: caller_id
- 3: location

#### Update Set Version
- 1: number
- 2: caller_id
- 3: u_caller_cell_2
- 4: location

### Related Links
- Return to Update Set

### Update Set Preview Problems
- Status =

#### Type
- Found a local update that is newer than this one
- Actions on selected rows...
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposition</td>
<td>Indicates when a collision is detected:</td>
</tr>
<tr>
<td></td>
<td>· Collision/Update, Collision/Insert, or Collision/Delete: the change is older than a change to the same object on the local instance.</td>
</tr>
<tr>
<td></td>
<td>· Update, Insert, or Delete: the change does not conflict with a change on the local instance.</td>
</tr>
<tr>
<td>File differences</td>
<td>Compares the most recent version of the object on the local instance with the version in the update set. Differences are marked with a color key. Deletions are highlighted in red, additions in green, and modifications in yellow.</td>
</tr>
<tr>
<td>Proposed action</td>
<td>Indicates how to handle the change when the update set is committed:</td>
</tr>
<tr>
<td></td>
<td>· Commit: Accept the change in the remote update. The default proposed action for every preview record is Commit, even if a newer update exists on the instance.</td>
</tr>
<tr>
<td></td>
<td>· Skip: Reject the change.</td>
</tr>
</tbody>
</table>

5. If necessary, resolve any problems listed in the **Update Problems** related list.
6. In the Proposed action field, select the action to take when committing the update set.
7. Click **Update** to save the action.
8. Repeat the process for every preview record.

**Resolve a preview problem in an update set**
Preview an update set to detect and resolve problems that may occur if you commit the updates on the local instance.
1. Navigate to **System Update Sets > Retrieved Update Sets**.
2. Open the update set record and scroll to the **Update Set Preview Problems** related list.
3. Review each problem description to determine the cause and resolve the problem.

Update Set Preview Problems

Missing Object

Example problem text: Could not find a record in sys_ui_policy referenced in this update.
Description: The object or a referenced object does not exist on the target instance. For example:

- An update modifies the form layout for a table that has not been created in the local instance.
- A UI policy action is included in the update set, but the parent UI policy is not.
- Resolution: Create another update set on the source instance to transfer the missing object to the local instance, or create the object on the local instance. Use these Available Actions to assist in problem resolution:
  - Find missing field or Find missing record: Opens a new window and searches the source instance for the missing field (dictionary entry) or record.
  - Find missing update: Opens a new window and searches the source instance for the update record that corresponds to the missing field or record.

Collision

Example problem text: Found a local update that is newer than this one.
Description: A change in the update set is older than a change to the same object on the local instance.
Resolution: Compare the two updates and determine which version to use. To use the version on the local instance, select Skip remote update. To use the version in the update set, select Accept remote update. Use these Available Actions to assist in problem resolution:
  - Compare with local: opens the preview record, which provides a comparison of the differences between the local version and the version in the update set.
  - Show local field or Show local record
  - Show local update

Uncommitted update

Example problem text: Could not find a table field (u_case.u_reference) referenced in this update, but did find it in another uncommitted update set.
Description: The object exists in another remote update set that has not been committed.
Resolution: Commit the other remote update set first or move this update to the other update set. Use these Available Actions to assist in problem resolution:
  - Show uncommitted update: opens the update record in the other remote update set.
  - Show uncommitted Update Set: open the other remote update set record.

Table to be deleted has data

Example problem text: Found a row in the table that is going to be deleted.
Description: One difference between table deletes and other metadata deletes is that the table data is lost when the table is deleted. (If the table is empty (no rows), then no problem is generated.)
Resolution: The problem must be ignored (delete will happen) or skipped (delete will not happen) before the update set can be committed. You can restore the table, but the restore does not bring back the data.
You are not allowed to delete system tables (ServiceNow tables) or tables outside your application scope.

**Application scope validation issue**
Description: The previewer identifies the following combination of states as a problem:
- The scope for the update set is not Global and
- The application is not found on the target instance and
- The application is not included with the update set and
- The application is not found on the ServiceNow Store.

Resolution: Transfer the update set only to instances that include the application scope or ensure that the update set includes the application.

**Conflict within a single batch**
Example problem text: This update has conflicts within the update set with the same name. Resolve the issue on the source system and re-preview or choose a specific update to use.
Description: Two or more update sets within the same batch have conflicting changes. The **Update Set Preview Problems** list contains a record for each update set with a conflicting change.
Resolution: Compare the conflicting update sets and determine which version to use. If you know which update set is the correct one to use, select the row for that set and click **Accept this collision**. Otherwise, click **Compare Collisions** to compare the conflicting update sets.
From this screen, you can compare any two of the conflicting update sets and choose the update set to commit.

Commit an update set
When you have previewed an update set and have resolved any issues, commit the update set. Committing an update set applies all changes to the instance and creates a local copy of the update set that contains an update record for every change.

2. Resolve any problems. You cannot commit an update set until all problems are resolved.
3. Click Commit Update Set.
   - Click Cancel to return to the preview and reevaluate the change. None of the updates are committed.
   - Click OK to skip the change and continue committing the changes that are marked as Commit.

If the update set contains one or more DELETEs for schema, the system displays a warning. The warning lists up to five updates that may contain problems. If more than five updates have potential problems, the system provides a link.

When the system successfully commits an update set, it displays a completion page.

4. (Recommended) Click Commit log on the confirmation page, or navigate to System Update Sets > Update log and filter for the update set name.
   - Look for warnings that contain the text unsafe edit. The system automatically skips any changes that result in data loss, such as changing the type of a field that contains data. You must manually make any of these changes, if necessary. Use caution when making changes that affect production data.
   - Look for errors that indicate which records failed to commit and why. Create a new update set to address those failures, if necessary.

5. (Recommended) When you are no longer working on the update set but do not want it transferred to another instance, navigate to System Update Sets > Local Update Sets and open the local update set record. Change the State to Ignore.

For completed update set on the production instance, you should always change the state to Ignore. This state ensures that the update set is not committed again when cloning the instance.

Back out an update set
You can back out changes to existing records for any committed update set.

Backing out an update set creates delete updates in the current update set. If you commit, back out, and then reapply a remote update set, errors appear in the previewer because the deleted updates are considered more recent changes and cause collisions.

**Warning:** Do not back out the Default update set. This action can damage the configuration of the instance.

The back out process reverses both record-level updates and changes to the dictionary. Some changes caused by a back-out can result in data loss. These are the expected results of the back-out process:
### Customer Update vs. Result of the back out action

<table>
<thead>
<tr>
<th>Customer Update</th>
<th>Result of the back out action</th>
</tr>
</thead>
<tbody>
<tr>
<td>A new table</td>
<td>The table is dropped from the database, deleting any data from it.</td>
</tr>
<tr>
<td>A new field</td>
<td>The field is dropped from the database, deleting any data from it.</td>
</tr>
<tr>
<td>A deleted field</td>
<td>The field is restored to the database, but the original data is lost.</td>
</tr>
<tr>
<td>A resized field</td>
<td>The field resize is reversed. If the field has been increased, data is truncated first to avoid errors.</td>
</tr>
<tr>
<td>A configured form</td>
<td>The form is reverted to its previous state.</td>
</tr>
<tr>
<td>A record is inserted</td>
<td>The record is deleted.</td>
</tr>
<tr>
<td>A record is deleted</td>
<td>The record is restored with its original data.</td>
</tr>
</tbody>
</table>

---

**Warning:** Backing out an update set that belongs to an update set batch may affect other update sets in the batch. For more information, see Back out batched update set.

1. Navigate to **System Update Sets > Retrieved Update Sets** or **System Update Sets > Local Update Sets**.
2. Open the update set record.

   **Note:** The currently selected application affects what options are available for the update set. Make sure you select the application, such as Global, that matches the contents of the update set.

3. Carefully review the contents of the update set and consider whether there will be problems if it is backed out.
   - If backing out an update set is not sufficient or will cause issues, then, instead, create and commit a new update set to reverse the customizations.
4. Click **Back Out**.
   - A Progress page displays actions, progress, and problems. Problems are changes in more recent update sets that affect the update set that is being backed out. The backout preview process generates a warning for each problem.
5. Resolve each problem before proceeding with the back out.
   - To keep the latest version, click **Decide to Keep Current**.
   - To back out to the previous version, click **Decide to Use Previous**.

All changes are reversed as described in the table. The current update set tracks all of the new changes that occur.

The update set and all associated update records are deleted. If needed, you can still navigate to the retrieved update set, preview it, and commit it again.

**Note:** If you commit, back out, and then reapply a remote update set, errors appear in the previewer because backing out an update set creates delete updates in the current update set. The deletes are considered more recent changes and cause collisions.
Cautions about deleting update sets

Deleting an update set is a bad practice. To revert a customization, back out the update set rather than deleting it.

Administrators can delete an update set only when it is not the current update set and it is empty (no sys_update_xml entries are associated with it). For example, after merging update sets, you might want to delete the original sets. This function is restricted by an access control rule (ACL) on the Update Set (sys_update_set) table.

ServiceNow strongly recommends that you do not delete sys_update_xml entries, because this action:

- Does not undo the updates.
- Removes any record of who applied the customizations.
- Removes the sys_update_xml entries associated with the update set, so customizations are overwritten when the instance is upgraded.

When you try to delete an update entry, a warning message appears. Click OK to confirm the deletion.

Update set batching

Batch update sets enable you to group update sets together so you can preview and commit them in bulk.

Dealing with multiple update sets can lead to problems, including committing update sets in the wrong order or inadvertently leaving out one or more sets. You can avoid these problems by grouping completed update sets into a batch.

The system organizes update set batches into a hierarchy. One update set can act as the parent for multiple child update sets. A given set can be both a child and parent, enabling multiple-level hierarchies. One update set at the top level of the hierarchy acts as the base update set.

When you preview or commit the base update set, you preview or commit the entire batch. The system determines the processing order, and checks for collisions, based on the dates the changes were recorded, and on their sequential ancestry. Their ancestries are the specific instances in which the changes in the update sets took place.

Example of batched update sets

The list of update set records reflects the batch hierarchy in the Parent and Batch Base columns. Diagram of batched update set hierarchy shows the hierarchy defined by the list in List of batched update sets.
List of batched update sets

Diagram of batched update set hierarchy
Create a batch update set

You include an update set in a batch by specifying another update set as its parent.

Role required: admin

Adding a WIP update set to a completed batch resets the batch base to WIP.

1. Navigate to System Update Sets > Local Update Sets.
2. Select the record for an update set that you want to include as a child in the batch.
3. On the Update Set record, navigate to the Parent field and select the update set to act as the parent.
4. Click Update.
   The system returns to the list of Update Sets. If the Parent column is visible, it shows the parent for the newly-created child.

Retrieve batched update sets

You retrieve a batch of update sets using the same process you would as for any individual update set.
Role required: admin

To retrieve a batch of update sets, follow the same process for the batch base as you would for any individual update set. The system will process the entire batch at once. For details, see Retrieve an update set.

**Preview a batch of update sets**

You can preview at once all the update sets belonging to a batch.

Role required: admin

You must have retrieved the update sets from the source instance.

1. Navigate to System Update Sets > Retrieved Update Sets
2. From the list of retrieved update sets, select the batch base for the batch you want to preview.
   - You cannot separately preview an update set that is a child in a batch. You must preview the entire batch by previewing the batch base. If necessary, you can remove the child update set from the batch by editing its record’s parent field.
3. Click Preview Update Set Batch.
4. If the system found problems, click the Preview Problems for Batch and resolve the problems as you normally would for any update set. When you have resolved all the problems, click Run Preview Again for Batch.

**Commit a batch of update sets**

You can commit at once all the update sets belonging to a batch.

Role required: admin

Before committing, you must have previewed the update sets from the source instance and resolved any collisions.

1. Navigate to System Update Sets > Retrieved Update Sets
2. From the list of retrieved update sets, select the batch base for the batch you want to preview.
   - You cannot separately commit an update set that is a child in a batch. You must commit the entire batch by committing the batch base. If necessary, you can remove the child update set from the batch by editing its record’s parent field.
3. Click Commit All Update Sets.

**Reorganize a batch of update sets**

You can remove an individual update set from the batch or change its parent.

Role required: admin

1. Navigate to System Update Sets > Local Update Sets.
2. Select the record for an update set that you want to move or remove as a child in the batch.
3. On the update set record, navigate to the Parent field and select the new update set to act as the parent. To remove the update set from the batch, delete any text from the Parent field and leave it blank.
4. Click Update.
   - The system returns to the list of update sets. If the Batch Base column is visible, it shows the parent for the newly-created child.

If the system property glide.update_set.auto_preview is set to true, the system automatically starts the preview process after the record is updated with a new parent. If this property is false, you
must start the process manually. For more information on the preview process, see Preview a batch of update sets.

Back out batched update set

Back out a batched update set by following the back out procedure for the base update set for the batch. You can also back out any child update set independently.

The following rules apply when backing out an update set that belongs to a batch:

- If the update set has a parent value, the system clears the parent value and treats the update set as an independent update set, or as a new batch base if it has any children.
- The system backs out the selected update set, plus any children of the backed-out update set.

Example of backing out a batched update set

If you back out Update Set 1.1 from the batch shown in List of batched update sets before backing out an update set, the result is the batch shown in List of batched update sets after backing out Update Set 1.1.
List of batched update sets before backing out an update set.
List of batched update sets after backing out Update Set 1.1

Hierarchical diagram of Update Set batch shows the hierarchy both before and after the back out. The red boxes show the update sets the system backs out if you back out Update Set 1.1.
Hierarchical diagram of Update Set batch

**APIs and scripts**

APIs are used in scripts to change functionality and to provide new features and applications to your instance.

<table>
<thead>
<tr>
<th>Explore</th>
<th>Set up</th>
<th>Administer</th>
</tr>
</thead>
</table>
| - API release notes  
- Domain separation and Script Debugger | - No set up is required to use scripts. | - Some APIs require a plug in to be activated. See the specific API description. |

<table>
<thead>
<tr>
<th>Use</th>
<th>Develop</th>
<th>Integrate</th>
</tr>
</thead>
</table>
| - Script information  
- Script debugger  
- JavaScript API reference  
- REST API reference | - Developer training  
- Developer documentation | - Web services |
Troubleshoot and get help

- Search the HI Knowledge Base for known error articles
- Contact ServiceNow Technical Support

JavaScript API reference

Use JavaScript APIs in scripts you write to change the functionality of applications, or when you create new applications.

AbstractDBObject

Provides common methods for classes based on records in the database.

Use this script include as a base class to create your own database object class.

AbstractDBObject - isValid()

Determines if the current database record is valid.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the database record is valid, false otherwise.</td>
</tr>
</tbody>
</table>

AbstractReconciler

Use this script include as a base class to create your own Discovery reconciler.

This is an abstract base class for reconciling newly discovered data in an XMLMemoryTable to a related list in the database. AbstractReconciler implements the common logic and helper methods for reconciling related lists for a particular CMDB item against discovered information present in a MemoryTable instance.

Most of this logic is very straightforward, but one bit may not be: the resolution of references to previously reconciled records. This feature works by collecting information while reconciling one related list, then passing that information on to the reconciler for another related list. For example, while switch partitions are being reconciled, information is gathered (in a map instance) that maps an interface number for a partition to the sys_id of the partition's record in the partition table. Then later when the switchports are being reconciled, a reference to the switch partition that contains the switchport can be resolved by using this map.

AbstractReconciler - getReconciliationKey()

Returns the string value of the field to use as a key to look up a record in the newly discovered data.
This value will be used to attempt to find a record in the discovered data, using the column returned by the `getReconciliationKey()` method.

Mandatory method, must be overridden in all concrete subclasses.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Value to use as a key into the discovered data.</td>
</tr>
</tbody>
</table>

**AbstractReconciler - getReconciliationField()**

Determines if the newly discovered data is different than the data already in the database for the current item.

This method is only invoked for items that have been rediscovered.

Mandatory method, must be overridden in all concrete subclasses.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the data has changed.</td>
</tr>
</tbody>
</table>

**AbstractReconciler - hasChanged()**

Returns true if the newly discovered data is different than the data already in the database for this item.

This method is only invoked for items that have been rediscovered.

Mandatory method, must be overridden in all concrete subclasses.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
_returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the discovered data is different than the data already in the database.</td>
</tr>
</tbody>
</table>

AbstractReconciler - process()

Reconciles the newly discovered data with the data already present in the database and resolves references to previously reconciled data.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AbstractReconciler - markAbsent()

Marks CIs as absent by setting the install status to "Absent"; other tables are marked by setting the (presumed present) field "absent" to true.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AbstractReconciler - markPresent()

Marks CIs as present by setting the install status to "Installed"; other tables are marked by setting the (presumed present) field "absent" to false.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
AbstractReconciler - readDatabaseFields()

Reads the current memory table record of newly discovered data and checks its validity.
Mandatory method, must be overridden in all concrete subclasses.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

AbstractReconciler - readDiscovered()

Reads the current memory table record of newly discovered data and checks its validity.
Mandatory method, must be overridden in all concrete subclasses.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the data is valid.</td>
</tr>
</tbody>
</table>

AbstractReconciler - resolveReferenceFields()

Reads the fields in the current database record into instance member fields.
Mandatory method, must be overridden in all concrete subclasses.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the data is valid.</td>
</tr>
</tbody>
</table>
AbstractReconciler - track(String sys_id)

Optional method to override in concrete subclasses that need to resolve reference fields.
This method is invoked after valid discovered data is read. Generally implementations will use some of this discovered data as a key into a map (passed into the concrete reconciler class when it was instantiated) that will return the sys_id of the database record holding the referenced information. However, implementations are not required to use this approach.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>The sys_id of the current database record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

AbstractReconciler - setDatabaseFields()

Sets the database fields for the current record to the values in the newly discovered information.
This method is invoked both for new database records and for existing database records when the discovered data has changed.
Mandatory method, must be overridden in all concrete subclasses.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

AbstractReconciler - setup()

Optional override for concrete subclasses that require special setup.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**AbstractScriptProcessor**

Base JavaScript processor class that other JavaScript processors extend.
Extend this class to create your own JavaScript processors.

**AbstractScriptProcessor - initialize(String request, String response, String process)**

Called by the Prototype JavaScript Framework during object construction.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>String</td>
<td>The request</td>
</tr>
<tr>
<td>response</td>
<td>String</td>
<td>The response</td>
</tr>
<tr>
<td>process</td>
<td>String</td>
<td>The process</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**AbstractScriptProcessor - process()**

Called by the Prototype JavaScript Framework during object processing. Do not call this method directly.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
**AbstractTimelineSchedulePage**

A helper class to facilitate working with the data displayed on timelines.

Processing the data displayed within a timeline starts by executing the corresponding function of the specified script include. Like other script includes, the language syntax is JavaScript and follows the default security constraints of this type of resource. However, because the different types of display options are complex, a helper class was created. To display a timeline, extend the `AbstractTimelineSchedulePage` class and pair it with its corresponding schedule page. At a minimum, extending classes should override the `getItems()` method, as this is the primary event handler for returning items to be displayed on the client.

The client processes the data that `AbstractTimelineSchedulePage` returns in two phases. The first phase makes the actual updates to the timeline. Immediately after (if configured), the second phase displays a success message box, error message box, or dialog message prompt. In phase one, the available options for manipulating data include:

- **Do Not Update Any Items** - This is the default behavior. Do not perform any of the remaining steps in phase one.
- **Update With Specific Items** - This is done using: `add()`.
- **Render The Timeline Using the `getItems()` Function** - This is done using: `setDoReRenderTimeline(true)`.

**Note:** If both `TimelineItems` are returned and `setDoReRenderTimeline` is set to `true`, the system will ignore the `setDoReRenderTimeline` property and explicitly show only the `TimelineItems` that were added via the `add()` function.

The available options in phase two include:

- **Do Not Display Any Message Boxes** - This is the default behavior.
- **Display a Success Dialog Box** - This is done using: `setStatusSuccess()`.
- **Display an Error Dialog Box** - This is done using: `setStatusError()`.
- **Display a Dialog Confirm Box** - This is done using: `setStatusPrompt()`.

**Note:**

A script include class that extends `AbstractTimelineSchedulePage` automatically receives all Uri parameters from the original Url whose prefix begins with `sysparm_timeline_`. To access the values of these, use:

```
this.getParameter("sysparm_timeline_VARIABLE");
```

This is useful if you need to display a schedule page from a dynamic element, such as from a context menu from a list. By passing in dynamic data via the Url the schedule page will auto-include these parameters inside the Ajax calls and therefore will be accessible inside the `AbstractTimelineSchedulePage` script includes.

**AbstractTimelineSchedulePage - add(Object ObjTimeline)**

Adds a `TimelineItem` object that will be returned to the client and appropriately displayed on the timeline.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjTimeline</td>
<td>Object</td>
<td>The TimelineItem object to add to the timeline.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

AbstractTimelineSchedulePage - addSeparator()

Adds a horizontal frame separator into the list of timeline items. All future items added via `add()` will be added into the subsequent timeline frame.

All future items added via `add()` will be added into the subsequent timeline frame.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
// Inside of a script include that extends
AbstractTimelineSchedulePage
this.addSeparator();
```

AbstractTimelineSchedulePage - elementMoveX(String spanSysId, String newStartDateTimeMs)

Event handler for when a timeline span moves horizontally.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spanSysId</td>
<td>String</td>
<td>The sys ID of the current span being adjusted.</td>
</tr>
</tbody>
</table>
### returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**AbstractTimelineSchedulePage - elementMoveY(String spanSysId, String itemSysId, String newItemSysId)**

Event handler for when a timeline span moves vertically.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spanSysId</td>
<td>String</td>
<td>The sys ID of the current span being adjusted.</td>
</tr>
<tr>
<td>itemSysId</td>
<td>String</td>
<td>The sys ID of the timeline item associated with the current span.</td>
</tr>
<tr>
<td>newItemSysId</td>
<td>String</td>
<td>The sys ID of the timeline item (a row) that the current span was dragged into.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**AbstractTimelineSchedulePage - elementMoveXY(String spanSysId, String itemSysId, String itemSysId, String newItemSysId, String newStartDateTimeMs)**

Event handler for when a timeline span moves both horizontally and vertically.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spanSysId</td>
<td>String</td>
<td>The sys ID of the current span being adjusted.</td>
</tr>
<tr>
<td>itemSysId</td>
<td>String</td>
<td>The sys ID of the timeline item associated with the current span.</td>
</tr>
</tbody>
</table>

| newStartDateTimeMs | String | The new start time of the span in milliseconds. Make sure to parse the string using parseInt() before performing any numerical comparisons. |
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### AbstractTimelineSchedulePage - elementSuccessor(String spanSysId, String newSuccSpanId)

Event handler for when a timeline relationship has been created between two spans.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spanSysId</td>
<td>String</td>
<td>The sys ID of the current span which will be a predecessor for the newly created relationship.</td>
</tr>
<tr>
<td>newSuccSpanId</td>
<td>String</td>
<td>The sys ID of the successor span from the relationship created.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>ReturnValue</td>
</tr>
</tbody>
</table>

#### AbstractTimelineSchedulePage - elementTimeAdjustStart(String spanSysId, String newStartDateTimeMs)

Event handler for when a timeline span's start date was modified.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spanSysId</td>
<td>String</td>
<td>The sys ID of the current span that is being adjusted.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
AbstractTimelineSchedulePage - newStartDateTimeMs

Name | Type | Description
--- | --- | ---
newStartDateTimeMs | String | The new start time of the span in milliseconds. Make sure to parse the string using `parseInt()` prior to performing any numerical comparisons.

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

AbstractTimelineSchedulePage - elementTimeAdjustEnd(String spanSysId, String newEndDateTimeMs)

Event handler for when a timeline span’s end date was modified.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spanSysId</td>
<td>String</td>
<td>The sys ID of the current span that is being adjusted.</td>
</tr>
<tr>
<td>newEndDateTimeMs</td>
<td>String</td>
<td>The new end time of the span in milliseconds. Make sure to parse the string using <code>parseInt()</code> prior to performing any numerical comparisons.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

AbstractTimelineSchedulePage - getItems()

Event handler for returning schedule items to display on the timeline.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
AbstractTimelineSchedulePage - inputBox(String strInputText)
Event handler for when a string was typed into the left pane input box.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strInputText</td>
<td>String</td>
<td>The text that was entered in the input box in the left pane.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

AbstractTimelineSchedulePage - itemMove(String itemSysId, String newItemSysId)
Event handler for when a timeline row item was moved and dragged into another row item.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemSysId</td>
<td>String</td>
<td>The sys ID of the timeline item associated with the current span.</td>
</tr>
<tr>
<td>newItemSysId</td>
<td>String</td>
<td>The sys ID of the timeline item (a row) that the current span was dragged into.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

AbstractTimelineSchedulePage - setDoReRenderTimeline(Boolean b)
Specifies whether or not to re-render all of the timeline items using the getItems() function.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If true, re-render the timeline by making a new event call to the server's getItems() handler.</td>
</tr>
</tbody>
</table>
AbstractTimelineSchedulePage - setPageTitle(String strTitle)

Specifies the text to display as the title of the timeline.

The page title can be set (and updated) from any interactive event; however, is recommended to be set during the `getItems()` event.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strTitle</td>
<td>String</td>
<td>The text to be displayed on the timeline title.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

AbstractTimelineSchedulePage - setStatusError(String strTitle, String strMessage)

Sets the current event request to an error status with a specified title and message to display in phase two of the GlideTimeline event processing.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strTitle</td>
<td>String</td>
<td>The text to be displayed in the dialog box title.</td>
</tr>
<tr>
<td>strMessage</td>
<td>String</td>
<td>The text to be displayed within the dialog box. The text can contain HTML formatting.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
The confirmation box displays an **OK** and **Cancel** button that each generate new events that will call the functions specified in the parameter arguments.

**Note:** The custom defined functions for **OK**, **Cancel**, and **Close** receive the same parameter arguments as those for the current event.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strTitle</td>
<td>String</td>
<td>The text to display in the dialog box. The text can contain HTML formatting.</td>
</tr>
<tr>
<td>strOkFunction</td>
<td>String</td>
<td>The function to call in the current script include class if the user clicks the <strong>OK</strong> button.</td>
</tr>
<tr>
<td>strCancelFunction</td>
<td>String</td>
<td>The function to call in the current script include class if the user clicks the <strong>Cancel</strong> button.</td>
</tr>
<tr>
<td>strCloseFunction</td>
<td>String</td>
<td>The function to call in the current script include class if the user clicks the <strong>Close</strong> button.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var MyTimelineScriptIncludeClass = Class.create();
MyTimelineScriptIncludeClass.prototype =
Object.extendsObject(AbstractTimelineSchedulePage, {

    getItems: function() {
        //...
    },

    elementTimeAdjustEnd: function(spanSysId, newEndDateTimeMs) {
        // Display a status prompt dialog box
        this.setStatusPrompt('Confirm Action', 'Are you sure you want to do that?',
            'this._myOkHandlerFunction',
            'this._myCancelHandlerFunction',
            'this._myCloseHandlerFunction');
    },

    _myOkHandlerFunction: function(spanSysId, newEndDateTimeMs) { // ... },

    _myCancelHandlerFunction: function(spanSysId, newEndDateTimeMs) { // ... },
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
AbstractTimelineSchedulePage - setStatusSuccess(String strTitle, String strMessage)
Sets the current event request to a success status with a specified title and message to display in phase two of the GlideTimeline event processing.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strTitle</td>
<td>String</td>
<td>The text to be displayed in the dialog box title.</td>
</tr>
<tr>
<td>strMessage</td>
<td>String</td>
<td>The text to be displayed within the dialog box. The text can contain HTML formatting.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

ActivityHistoryRecord
A data object that is a member of the WorkflowModelManager class; ActivityHistoryRecord methods facilitate interacting with the WorkflowModelManager object.

The ActivityHistoryRecord is basically a copy of the data from a GlideRecord on table wf_history. There are some helper methods defined for this class as well. There are no accessors or mutator to the simple data elements of this class.

ActivityHistoryRecord - addArrivedActivityToJoin(String Array activityIds)
Changes the arrivedState of already known ids to true.

As soon as this method changes the arrivedState of known ids to true, it tests to see if the join is satisfied. If it is, the model walk can continue; if not, it should continue to the next appropriate transition. This is called during playback, as expected wf_activities flow to the join.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activityIds</td>
<td>String Array</td>
<td>Array of wf_activity.sys_ids that are headed towards join.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ActivityHistoryRecord - addJoinFromActivityIds(String Array activityIds)**

Seeds all the expected `wf_activity` sys_ids that are expected to pass through this join and sets their `arriveState = false`.

This state is flipped to true as each expected activity transitions to this instance of the Join. This method is called as part of building the cached model in memory, and is not called during playback.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activityIds</td>
<td>String Array</td>
<td>Array of <code>wf_activity.sys_ids</code> that are headed towards join.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ActivityHistoryRecord - addTransition(Object InTransition)**

As model is cached by `_getExecutedTransitions()`, this method adds transitions that have gone FROM this activity towards the .to activity.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InTransition</td>
<td>Object</td>
<td>ExecutedTransition Javascript object to add to this activity.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ActivityHistoryRecord - debugDump()**

Prints debug information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ActivityHistoryRecord - doesJoinContainActivity()
Tests the incoming wf_activity sys_id, presumed to be seeded in this history record. If it is, it is tested to see if it is an activity that would come through this join.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

ActivityHistoryRecord - getLogString()
Returns the log string from the history record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The log string.</td>
</tr>
</tbody>
</table>

ActivityHistoryRecord - getSatisfiedJoinActivities()
Returns the sys_ids of the history records that transition to this Join activity that have already come through.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Array</td>
<td>Array of sys_ids of the history records that transition to this Join activity.</td>
</tr>
</tbody>
</table>

**ActivityHistoryRecord - getTransitionCount()**

Returns the transition count for this activity.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Number of transitions from this activity.</td>
</tr>
</tbody>
</table>

**ActivityHistoryRecord - getUnSatisfiedJoinActivities()**

Returns the sys_ids of the history records that transition to this Join activity that the join is still waiting for.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Array</td>
<td>Array of sys_ids of the history records that transition to this Join activity,</td>
</tr>
</tbody>
</table>

**ActivityHistoryRecord - isARollback()**

Determines if this activity is one of the Rollback activities.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True, if activity is one of the Rollback activities.</td>
</tr>
</tbody>
</table>

ActivityHistoryRecord - isIdADestination(String ahrSys_id)

Determines if the sys_id passed in is a destination of any of the transitions associated with this instance of an ActivityHistoryRecord.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ahrSys_id</td>
<td>String</td>
<td>Activity history sys_id from table wf_history.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True, if the sys_id passed in is a destination of any of the transitions.</td>
</tr>
</tbody>
</table>

ActivityHistoryRecord - isJoin( )

Determines if this activity is a join.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True, if the activity is &quot;Join&quot;.</td>
</tr>
</tbody>
</table>

ActivityHistoryRecord - isJoinSatisfied( )

Examines the value of the arriveState.

Called during playback when the transition to sys_id of a transition points to an activity that is a join.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Boolean</td>
<td>True, if the join is satisfied.</td>
<td></td>
</tr>
</tbody>
</table>

**ActivityHistoryRecord - isJoinWaitingForActivity(String Array activityIds)**

Tests the incoming wf_activity sys_id, presumed to be seeded in this history record.

If it is, it is tested to see if it has already been through; if it has, then this join is not waiting on this activity and it should look further downstream for the one that is waiting. This test is called in sequence as the model is walked, so it does presume in the instances of rollbacks and loops. If the sys_id is in here and the flag is **true**, this waiting Join is further up the sequence. This sort of check is required as the history records of Joins are removed from the history table, and so it is possible for a record to be in the transition table with no match in the corresponding history table, but is still an appropriate match for an earlier executed instance of the same Join.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activityIds</td>
<td>String Array</td>
<td>Array of wf_activity.sys_ids that are headed towards join.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True, if the activity has already been through.</td>
</tr>
</tbody>
</table>

**ActivityHistoryRecord - isRolledBack( )**

Determines if this workflow was rolled back.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True, if this workflow was rolled back.</td>
</tr>
</tbody>
</table>

**ActivityHistoryRecord - isTurnstile( )**

Determines if this activity is a turnstile.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True, if this activity is a turnstile.</td>
</tr>
</tbody>
</table>

AgentMetrics

Processes inbound MID server metrics. Updates these tables.
- ECC Agent Counter Metric
- ECC Agent Memory Metric
- ECC Agent RGR Metric
- ECC Agent Scalar Metric

Use these methods in server scripts.

AgentMetrics - AgentMetrics()

Creates an instance of AgentMetrics.

AgentMetrics - handleCounter(String metric, String name, String midId)

Handles counter metrics from the agent by putting them into an array that can be used to update the ECC Agent Counter Metric table (ecc_agent_counter_metric).

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric</td>
<td>String</td>
<td>The counter object from the agent</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The device name</td>
</tr>
<tr>
<td>midId</td>
<td>String</td>
<td>The MID server sys_id</td>
</tr>
</tbody>
</table>
AgentMetrics - handleMemory(String metric, String name, String midId)

Updates the memory metrics for the given agent.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric</td>
<td>String</td>
<td>The memory object from the agent</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The device name</td>
</tr>
<tr>
<td>midId</td>
<td>String</td>
<td>The MID server sys_id</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

AgentMetrics - handleRGR(String metric, String name, String midId)

Handles metric information from a remote glide record by putting it into an array.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric</td>
<td>String</td>
<td>The remote object from the agent</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The device name</td>
</tr>
<tr>
<td>midId</td>
<td>String</td>
<td>The MID server sys_id</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
AgentMetrics - `handleRGR(String metric, String name, String midId)`

Handles scalar values from the agent by putting them into an array.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric</td>
<td>String</td>
<td>The counter object from the agent</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The device name</td>
</tr>
<tr>
<td>midId</td>
<td>String</td>
<td>The MID server sys_id</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var amtc = new AgentMetrics();
amtc.handleRGR('remote_obj', 'server_name', '1098888bbb9483abcd89981ffaeff');
```

AgentMetrics - `handleScalar(String metric, String name, String midId)`

Updates the specified metrics table with the new metric information.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The metrics table to update</td>
</tr>
<tr>
<td>values</td>
<td>String</td>
<td>The object containing the values to add to the table</td>
</tr>
<tr>
<td>metric</td>
<td>String</td>
<td>The metric type to add to the table</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var amtc = new AgentMetrics();
amtc.handleScalar('scalar', 'server_name', '1098888bbb9483abcd89981ffaeff');
```
### ServiceNow, Kingston, Now Platform Custom Business Applications

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>midID</td>
<td>String</td>
<td>The MID server sys_id</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var amtc = new AgentMetrics();
amtc.handleMemory('memory', 'server_name', '1098888bbb9483abcd89981ffaeff');
amtc.updateMetric('ecc_agent_memory_metric', 'm_values', 'memory', '1098888bbb9483abcd89981ffaeff');
```

**APCHandler**

Handles SNMP classification and identification for UPSs, environmental monitors, and PDUs made by APC.

Use this API for SNMP classification and identification.

**APCHandler - classifyAndIdentify()**

Classifies and identifies APC devices.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ApcPduOutletReconciler**

Reconciles outlets for power distribution units.

Use this API for SNMP-related discovery.

**ApcPduOutletReconciler - getReconciliationField()**

Returns the reconciliation field.
### ServiceNow Custom Business Applications

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The reconciliation field</td>
</tr>
</tbody>
</table>

#### `ApcPduOutletReconciler - getReconciliationKey()`

This method returns the reconciliation key.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The reconciliation key</td>
</tr>
</tbody>
</table>

#### `ApcPduOutletReconciler - hasChanged()`

This method determines if the outlet state has changed.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the outlet state has changed; otherwise, false.</td>
</tr>
</tbody>
</table>

#### `ApcPduOutletReconciler - readDatabaseFields()`

This method reads the outlet database fields.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ApcPduOutletReconciler - readDiscovered()
Reads the discovered outlet database fields.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

ApcPduOutletReconciler - setDatabaseFields()
Sets the outlet database fields.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

ArrayUtil

ArrayUtil API is a script include with useful functions for working with JavaScript arrays.
These methods are available to any server-side script.

ArrayUtil - concat(Array parent, Array child)
Merges two arrays.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>Array</td>
<td>An array to merge</td>
</tr>
<tr>
<td>child</td>
<td>Array</td>
<td>An array to merge</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Array</td>
<td>An array of elements from both input arrays. Duplicates are not removed.</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var arrayUtil = new ArrayUtil();
var a1 = new Array("a", "b", "c");
var a2 = new Array("c", "d", "e");

gs.print("concat a1, a2: " + arrayUtil.concat(a1, a2));
```

Output: concat a1, a2: a,b,c,c,d,e

**ArrayUtil - contains(Array array, Object element)**

Searches the array for the element. Returns true if the element exists in the array, otherwise returns false.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Array</td>
<td>The array to search</td>
</tr>
<tr>
<td>element</td>
<td>Object</td>
<td>The element to search for</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the element is in the array, false otherwise.</td>
</tr>
</tbody>
</table>

```javascript
var arrayUtil = new ArrayUtil();
var a1 = new Array("a", "b", "c");

gs.print("Contains b: " + arrayUtil.contains(a1, "b");
gs.print("Contains x: " + arrayUtil.contains(a1, "x");
```

Output:

Contains b: true
Contains x: false

**ArrayUtil - convertArray(Object a)**

Converts an object to an array.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Object</td>
<td>The object to be converted.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array created from the object.</td>
</tr>
</tbody>
</table>

ArrayUtil - diff(Array a, Array b)

Finds the differences between two or more arrays. Any number of arrays can be provided as parameters.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Array</td>
<td>An array</td>
</tr>
<tr>
<td>b</td>
<td>Array</td>
<td>An array</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>Returns an array of items from array a that were not found in either array b or c, or other input arrays. Duplicates are removed from the result.</td>
</tr>
</tbody>
</table>

```javascript
var arrayUtil = new ArrayUtil();
var a1 = new Array("a", "b", "c");
var a2 = new Array("c", "d", "e");
gs.print(arrayUtil.diff(a1, a2));
```

Output: a,b

ArrayUtil - ensureArray(Object object)

Returns an array from the object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Object</td>
<td>The object from which to create an array.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array created from the object.</td>
</tr>
</tbody>
</table>

**ArrayUtil - indexOf(Array array, Object element)**

Searches the array for the element. Returns the element index if found, -1 otherwise.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Array</td>
<td>The array to search</td>
</tr>
<tr>
<td>element</td>
<td>Object</td>
<td>The element to search for</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The index where the element was found, -1 otherwise.</td>
</tr>
</tbody>
</table>

```javascript
var arrayUtil = new ArrayUtil();
var a1 = new Array("a", "b", "c");
gs.print("indexOf b: " + arrayUtil.indexOf(a1, "b");
gs.print("indexOf x: " + arrayUtil.indexOf(a1, "x");
```

Output:

```
indexOf b: 1
indexOf x: -1
```

**ArrayUtil - indexOf(Array array, Object element, Number startIndex)**

Searches the array for the element starting with the startIndex element. Returns the element index.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Array</td>
<td>The array to search</td>
</tr>
<tr>
<td>element</td>
<td>Object</td>
<td>The element to search for</td>
</tr>
<tr>
<td>startIndex</td>
<td>Number</td>
<td>The index to begin the search</td>
</tr>
</tbody>
</table>
ArrayUtil - intersect(Array a, Array b)

Finds the elements present in all arrays.
Any number of arrays can be provided as parameters.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Array</td>
<td>An array</td>
</tr>
<tr>
<td>b</td>
<td>Array</td>
<td>An array</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of elements from array a that were found in all of the other input arrays. Duplicates are removed.</td>
</tr>
</tbody>
</table>

```javascript
var arrayUtil = new ArrayUtil();
var a1 = new Array("a", "b", "c");
var a2 = new Array("c", "d", "e");
gs.print(arrayUtil.intersect(a1, a2));
```

Output: c

ArrayUtil - union(Array a, Array b)

Merges two or more arrays.
Any number of arrays can be provided as parameters.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Array</td>
<td>An array</td>
</tr>
<tr>
<td>b</td>
<td>Array</td>
<td>An array</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of items from all the input arrays. Duplicates are removed.</td>
</tr>
</tbody>
</table>

```javascript
var arrayUtil = new ArrayUtil();
var a1 = new Array("a", "b", "c");
var a2 = new Array("c", "d", "e");
gs.print(arrayUtil.union(a1, a2));
```

Output: `a,b,c,d,e`

ArrayUtil - unique(Array a)

Removes duplicate items from an array.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Array</td>
<td>The array to check for duplicate elements.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of unique items from the input array.</td>
</tr>
</tbody>
</table>

```javascript
var arrayUtil = new ArrayUtil();
var a1 = new Array("a", "b", "c", "c", "b");
gs.print(arrayUtil.unique(a1));
```

Output: `a,c,b`

AssessmentCreation

The AssessmentCreation class provides methods that administrators can use to generate assessments and surveys.

There are multiple ways to call the createAssessments method. You must set the `typeID` parameter in all cases to identify a metric type or survey definition.

Some of the factors that influence how the method behaves include:

- Whether the target is an assessment or survey
- Whether the assessment schedule type is on-demand or scheduled
- Survey schedule periods
Parameters provided

For example, when the method is called with only the typeID parameter set, the method searches for stakeholders or survey users to send invitations to. In contrast, when the method is called with the typeID and userID parameters set, it attempts to send invitations to the specified users.

**AssessmentCreation - CreateAssessments (String typeID, String sourceRecordID, String userID)**

Creates one or more assessments or surveys for the specified metric type or survey definition.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>typeID</td>
<td>String</td>
<td>The sys_id of the metric type or survey definition for which to generate assessments or surveys.</td>
</tr>
<tr>
<td>sourceRecordID</td>
<td>String</td>
<td>One or more comma-separated sys_ids of records to include in the assessments generated. There must be an assessable record associated with the specified metric type for each source record. If this parameter is left blank, the assessments generated include all assessable records for the specified type. This parameter is for use with assessments only.</td>
</tr>
<tr>
<td>userID</td>
<td>String</td>
<td>One or more comma-separated sys_ids of users to which to send assessment or survey instances. If this parameter is left blank, the assessment stakeholders or survey users receive instances. This parameter is required for on-demand assessments.</td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>If the method is successful, it returns a comma-separated string such as 7a2ad2253784200044e0bfc8bcebe5de1, 10, b7e8b5d4c0a80169008b49e468920048 (Instance ID, number of instances, Group ID) with the following values:</td>
</tr>
<tr>
<td></td>
<td>· instanceID: Sys_id of the assessment or survey instance created, if there is only one. Sys_id of one of the instances, if there are multiple. If there are multiple instances, use the group ID to retrieve data.</td>
</tr>
<tr>
<td></td>
<td>· #instances: Number of assessment or survey instances created.</td>
</tr>
<tr>
<td></td>
<td>· groupID: Sys_id of the assessment group that contains the assessment or survey instances created.</td>
</tr>
<tr>
<td></td>
<td>If the method is unable to generate assessments or surveys, it returns one of the error codes.</td>
</tr>
<tr>
<td><strong>Error Codes</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· issued - Indicates that the <code>userID</code> or <code>sourceRecordID</code> parameter should not have been used. When the <code>typeID</code> value represents a metric type with the Schedule type field set to Scheduled, the <code>typeID</code> parameter is the only parameter that may be used.</td>
</tr>
<tr>
<td></td>
<td>· draft_state - Indicates that the <code>typeID</code> value is a metric type or survey definition that is not published. Publish the metric type or survey definition.</td>
</tr>
<tr>
<td></td>
<td>· invalidtype - Indicates that there is no existing metric type that matches the <code>typeID</code> value.</td>
</tr>
<tr>
<td></td>
<td>· inactive - Indicates that the <code>typeID</code> value is a metric type or survey definition that is not active. Navigate to the metric type or survey definition and select the Active check box.</td>
</tr>
<tr>
<td></td>
<td>· not_available - Indicates that the <code>typeID</code> value is a metric type or survey definition that has no metrics or survey questions. Add at least one metric or survey question.</td>
</tr>
<tr>
<td></td>
<td>· notyet - Indicates that at least one of the users in the <code>userID</code> value is assigned a survey instance for the specified survey definition, that has yet to expire. The error code returns the expiration date.</td>
</tr>
<tr>
<td></td>
<td>· wip - Indicates that at least one of the users in the <code>userID</code> value is already assigned a survey instance for the survey.</td>
</tr>
<tr>
<td></td>
<td>· not_authorized - Indicates that at least one of the users in the <code>userID</code> value is not authorized to take the survey. Verify that one of the</td>
</tr>
</tbody>
</table>
**AutomationException**

Provides generic exception wrapping for automation.

Include this in automation code to implement error handling.

**AutomationException - getMessage()**

Retrieves the error message.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td></td>
<td>The error message</td>
</tr>
</tbody>
</table>

```javascript
var ae = new AutomationException("This is an error message");
gs.print(ae.getMessage());
```

Output: This is an error message

**cabrillo**

Cabrillo JS is a client side JavaScript API for accessing capabilities inside ServiceNow native mobile applications.

Use Cabrillo JS to build a custom mobile experience in ServiceNow native mobile applications.

Cabrillo JS provides APIs to use native device capabilities, native mobile UI, and other ServiceNow mobile functionality.

Cabrillo JS is available in AngularJS as an injectable parameter. For example, you can inject Cabrillo JS in a Service Portal widget client script.

```javascript
function(cabrillo) {
    var c = this;
    // Used to determine if Cabrillo is executing in ServiceNow's native mobile apps.
    c.isNative = cabrillo.isNative();
}
```

Cabrillo JS APIs are only supported when executing in ServiceNow native mobile applications. Use the `cabrillo.isNative()` API to determine if the script is executing in a native context.

**Note:** Cabrillo JS APIs are unavailable in GlideForm client scripts.

**cabrillo - isNative()**

Boolean value indicating if Cabrillo JS is executing a ServiceNow native mobile application.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if Cabrillo JS is running in a native context; otherwise, false.</td>
</tr>
</tbody>
</table>

```javascript
var isNative = cabrillo.isNative();
```

Cabrillo JS constants

These are the constants used by Cabrillo JS.

Cabrillo JS button styles

Constants to use when setting the style of a button.

Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cabrillo.viewLayout.MORE_MENU_BUTTON_STYLE</td>
<td>String</td>
<td>Indicates that the button appears in the navigation bar’s overflow button menu.</td>
</tr>
</tbody>
</table>

Cabrillo JS close button styles

Constants to use when setting the close button style of modal interfaces.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cabrillo.modal.CLOSE_BUTTON_STYLE_CANCEL</td>
<td>String</td>
<td>The modal has a localized cancel button to close the modal.</td>
</tr>
<tr>
<td>cabrillo.modal.CLOSE_BUTTON_STYLE_CLOSE</td>
<td>String</td>
<td>The modal has a localized close button to close the modal.</td>
</tr>
<tr>
<td>cabrillo.modal.CLOSE_BUTTON_STYLE_DONE</td>
<td>String</td>
<td>The modal has a localized done button to close the modal.</td>
</tr>
</tbody>
</table>

Cabrillo JS modal presentation styles

Constants to use when setting the presentation style of modal interfaces.
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cabrillo.modal.MODAL_PRESENTATION_STYLE_FULLSCREEN</code></td>
<td>String</td>
<td>The modal is presented full screen.</td>
</tr>
<tr>
<td><code>cabrillo.modal.MODAL_PRESENTATION_STYLE_FORM_SHEET</code></td>
<td>String</td>
<td>The modal is presented as a form sheet, but may adapt to full screen on small devices.</td>
</tr>
</tbody>
</table>

### Cabrillo JS data types

These are common data types used by Cabrillo JS APIs.

#### Cabrillo.Attachment

Defines an attachment.

#### Property

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>content_type</code></td>
<td>String</td>
<td>The attachment content type.</td>
</tr>
<tr>
<td><code>ext</code></td>
<td>String</td>
<td>The attachment file extension.</td>
</tr>
<tr>
<td><code>file_name</code></td>
<td>String</td>
<td>The full file name of the attachment including the extension.</td>
</tr>
<tr>
<td><code>sys_created_by</code></td>
<td>String</td>
<td>The user name of the user that created the attachment.</td>
</tr>
<tr>
<td><code>sys_id</code></td>
<td>String</td>
<td>The attachment record sys_id.</td>
</tr>
<tr>
<td><code>sys_updated_on</code></td>
<td>String</td>
<td>The date the attachment record was updated.</td>
</tr>
<tr>
<td><code>table_name</code></td>
<td>String</td>
<td>Name of the table containing the record to which the attachment is attached.</td>
</tr>
<tr>
<td><code>table_sys_id</code></td>
<td>String</td>
<td>The sys_id of the record to which the attachment is attached.</td>
</tr>
<tr>
<td><code>thumbnail</code></td>
<td>String</td>
<td>The URL for the image thumbnail of the attachment. Only applies to image attachments.</td>
</tr>
</tbody>
</table>

### Cabrillo.Button

Defines a button. Different interface contexts may support a subset of a button’s properties.
## Property

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td>Title of the button. The string should be localized.</td>
</tr>
<tr>
<td>enabled</td>
<td>Boolean</td>
<td>Indicates whether the button is enabled.</td>
</tr>
<tr>
<td>badgeCount</td>
<td>Number</td>
<td>Number value to display on the button's badge.</td>
</tr>
<tr>
<td>backgroundColor</td>
<td>String</td>
<td>Background color of the button or badge. A string containing hex, RGB, HSL and x11 named color.</td>
</tr>
<tr>
<td>textColor</td>
<td>String</td>
<td>The text color of the badge or button. A string containing hex, RGB, HSL and x11 named color.</td>
</tr>
<tr>
<td>buttonStyle</td>
<td>String</td>
<td>Used to customize the style of the button.</td>
</tr>
<tr>
<td>imageName</td>
<td>String</td>
<td>Specifies an image. Possible values are action, add, article-document, book, bookmarks, calendar, camera, cart, cart-full, catalog, cog, comment, compose, configuration, console, dashboard, database, directions, drawer, edit, fast-forward, filter, form, help, home, hr, image, info, key, label, lightbulb, location, locked, mail, mobile, new-ticket, notification-bell, organize, paperclip, pause, play, refresh, reply, rewidn, script, search, sort-ascending, trash, tree, tree-right, user, user-group, view.</td>
</tr>
</tbody>
</table>

---

## Cabrillo.ListSelection

Describes a selected list item.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>String</td>
<td>The value of the selected item. Default is the sys_id of the record, or the value of a custom reference key.</td>
</tr>
<tr>
<td>displayValue</td>
<td>String</td>
<td>The display value of the selected item.</td>
</tr>
</tbody>
</table>
Cabrillo.Location
Contains geographical information, latitude, longitude, and direction.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>altitude</td>
<td>Number</td>
<td>The altitude in meters.</td>
</tr>
<tr>
<td>coordinate</td>
<td>Cabrillo.LocationCoordinate</td>
<td>The geographical coordinates.</td>
</tr>
<tr>
<td>direction</td>
<td>Number</td>
<td>Measured in degrees beginning at due north.</td>
</tr>
<tr>
<td>speed</td>
<td>Number</td>
<td>Measured in meters per second.</td>
</tr>
<tr>
<td>timestamp</td>
<td>String</td>
<td>The date that the location was last updated.</td>
</tr>
</tbody>
</table>

Cabrillo.LocationCoordinate
Contains a geographical coordinate.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>latitude</td>
<td>Number</td>
<td>The latitude in degrees.</td>
</tr>
<tr>
<td>longitude</td>
<td>Number</td>
<td>The longitude in degrees.</td>
</tr>
</tbody>
</table>

Cabrillo.ModalResponse
Contains the result passed from a modal window that was dismissed.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>results</td>
<td>Object</td>
<td>The result from the dismissed modal window.</td>
</tr>
</tbody>
</table>

Cabrillo.NavigationRequest
Specifies a request to navigate to a list or record.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>The table to navigate to. Required for both list and record navigation.</td>
</tr>
<tr>
<td>sysId</td>
<td>String</td>
<td>The sys_id of the record to navigate to. Required for record navigation.</td>
</tr>
</tbody>
</table>
### Property

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>String</td>
<td>An encoded query for the requested list. Can be used to seed values for record navigation requests.</td>
</tr>
<tr>
<td>view</td>
<td>String</td>
<td>The list or record view.</td>
</tr>
</tbody>
</table>

### Cabrillo.Rect

Contains the location and dimensions of a rectangle.

#### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Number</td>
<td>The x origin of the rectangle.</td>
</tr>
<tr>
<td>y</td>
<td>Number</td>
<td>The y origin of the rectangle.</td>
</tr>
<tr>
<td>width</td>
<td>Number</td>
<td>The width in points.</td>
</tr>
<tr>
<td>height</td>
<td>Number</td>
<td>The height in points.</td>
</tr>
</tbody>
</table>

### Cabrillo.attachments

The name space for Cabrillo JS attachment functions. This enables adding and viewing attachments.

#### cabrillo.attachments - addFile(String tableName, String sysId, Object params, String options)

Presents a document picker and uploads the selected file.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Table name of the record to which to attach the attachment.</td>
</tr>
<tr>
<td>sysId</td>
<td>String</td>
<td>The sys_id of the record to which to attach the attachment.</td>
</tr>
<tr>
<td>params</td>
<td>Object</td>
<td>Reserved for future use. Set to null.</td>
</tr>
<tr>
<td>options</td>
<td>String</td>
<td>Reserved for future use. Set to null.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>promise</td>
<td>If successful a Cabrillo.Attachment object. If the operation fails, an error.</td>
</tr>
</tbody>
</table>

```javascript
var table = 'incident';
var sysID = 'a9e30c7dc61122760116894de7bcc7bd';

cabrillo.attachments.addFile(table,
   sysID,
   null,
   null
).then(function(attachment) {
   if (attachment) {
      console.log('Added a new file.', attachment);
   } else {
      console.log('User cancelled adding an attachment.');
   }
}, function(error) {
   console.log('Failed to attach new file.', error);
});
```

cabrillo.attachments -viewFile(Cabrillo.Attachment attachment, Cabrillo.Rect sourceRect, String sourceBase64Image)

Presents a document picker and uploads the selected file.

**Note:** Scaling using a sourceRect parameter with a sourceBase64 image is only supported on iOS. Android ignores these parameters and opens the image without a scaling animation.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attachment</td>
<td>Cabrillo.Attachment</td>
<td>Describes the attachment to view.</td>
</tr>
<tr>
<td>sourceRect</td>
<td>Cabrillo.Rect</td>
<td>Optional. Describes the source rectangle of the image to scale up.</td>
</tr>
<tr>
<td>sourceBase64Image</td>
<td>String</td>
<td>Optional. A base64 representation of the source image to scale up.</td>
</tr>
</tbody>
</table>
// A Cabrillo.Attachment dictionary to view
var attachment = {
    sys_id: '8e99daa3ff133100ba13ffffff2d',
    content_type: 'image/jpeg',
    path: '8e99daa3ff133100ba13ffffff2d.iix'
};

cabrillo.attachments.viewFile(attachment,
    null,
    null
).then(function() {
    // It worked. Nothing to do here.
}, function(error) {
    console.log('Failed to view file.', error);
});

To scale an image that was tapped into a native image viewer, the `viewFile()` method accepts optional arguments for the image's rectangle on the page as well as a base64 encoded thumbnail of the image. The thumbnail is scaled into the full size image with an animation.

// Grab image metadata from an image that was tapped
var imageMetadata = imageMetadataFromEvent(event);

// Optional rect of image on page
var imageRect = imageMetadata.rect;

// Optional base64 encoded image to scale up into native viewer
var base64EncodedImage = imageMetadata.base64;

// A Cabrillo.Attachment dictionary to view
var attachment = {
    sys_id: '8e99daa3ff133100ba13ffffff2d',
    content_type: 'image/jpeg',
    path: '8e99daa3ff133100ba13ffffff2d.iix'
}

cabrillo.attachments.viewFile(attachment,
    imageRect,
    base64EncodedImage
).then(function() {
    // It worked. Nothing to do here.
}, function(error) {
    console.log('Failed to view file.', error);
});

cabrillo.camera
Cabrillo JS camera functions to provide camera access for bar code scanning.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>promise</td>
<td>If successful, is undefined; otherwise an error.</td>
</tr>
</tbody>
</table>
cabrillo.camera - getBarcode()

Provides a standard interface for scanning bar codes. Use this to present a bar code scanner and capture scanned result.

Supported bar code types.
- 2D bar codes: QR Code, Data Matrix, PDF-417, AZTEC

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>promise</td>
<td>Resolves to a bar code string, or if the operation fails, an error.</td>
</tr>
</tbody>
</table>

cabrillo.camera.getBarcode().then(function(barcode) {
  if (barcode) {
    console.log('Scanned barcode.', barcode);
  } else {
    console.log('User cancelled scanning barcode.');
  }
}, function(error) {
  console.log('Failed to scan barcode.', error);
});

cabrillo.geolocation

Cabrillo JS functions to provide access to the device’s location services.

cabrillo.geolocation - getCurrentLocation()

Returns the current location of the device.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**cabraillo.geolocation**

```javascript
cabraillo.geolocation.getCurrentLocation().then(function(location)
{
    console.log('Current location:', location);
}, function(error)
{
    console.log('Failed to get current location.', error);
});
```

**cabraillo.list**

Cabraillo JS functions to provide native UI for selecting items from a list.

**cabraillo.list - selectItem(String title, String tableName, String query, Cabriollo.ListSelection selectedItem, Object params)**

Presents a list to the user to select a single item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td>The title of the list interface.</td>
</tr>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table name from which to select the item.</td>
</tr>
<tr>
<td>query</td>
<td>String</td>
<td>The encoded query to use as a filter for the list.</td>
</tr>
<tr>
<td>selectedItem</td>
<td>Cabriollo.ListSelection</td>
<td>The currently selected item. (Optional)</td>
</tr>
<tr>
<td>params</td>
<td>Object</td>
<td>Reserved for future use. Always specify null.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>promise</td>
<td>If successful a Cabriollo.ListSelection object, otherwise an error.</td>
</tr>
</tbody>
</table>

Select a single item from a list.

```javascript
cabraillo.list.selectItem('Select Incident',
    'incident',
    'active=true',
    null,
    null
```
Change a selection using a list when the list already has an active selection. Use the selectedItem parameter to specify the selected item.

```javascript
var selectedItem = {
  value: '681b365ec0a80164000fb0b05854a0cd',
  displayValue: 'Abel Tuter'
};

cabrillo.list.selectItem('Select User',
  'sys_user',
  'active=true',
  selectedItem,
  null
).then(function(selection) {
  if (selection) {
    console.log('Selected item in list.', selection);
  } else {
    console.log('User cancelled selecting an item.');
  }
}, function(error) {
  console.log(error);
});
```

$cabrillo.list - selectItems(String title, String tableName, String query, Array selectedItems, Object params)$

Presents a list to the user to select multiple items.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td>The title of the list interface.</td>
</tr>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table name from which to select the items.</td>
</tr>
<tr>
<td>query</td>
<td>String</td>
<td>The encoded query to use as a filter for the list.</td>
</tr>
<tr>
<td>selectedItems</td>
<td>Array of Cabrillo.ListSelection objects</td>
<td>The currently selected items. (Optional)</td>
</tr>
<tr>
<td>params</td>
<td>Object</td>
<td>Reserved for future use. Always specify null.</td>
</tr>
</tbody>
</table>
Select multiple items from a list.

```javascript
const selectItems = function() {
  cabrillo.list.selectItems('Select Incidents',
    'incident',
    'active=true',
    null,
    null
  ).then(function(selections) {
    if (selections) {
      console.log('Selected items in list.', selections);
    } else {
      console.log('User cancelled selecting items.');
    }
  }, function(error) {
    console.log(error);
  });
}
```

Change multiple list selections when the list has active selections. Use the `selectedItems` parameter to specify the active selections.

```javascript
const selectedItems = [
  {value: '681b365ec0a80164000fb0b05854a0cd',
    displayValue: 'Abel Tuter'},
  {value: '0a826bf03710200044e0bfc8bcbe5d7a',
    displayValue: 'Adela Cervantsz'}
];

const changeActiveSelections = function() {
  cabrillo.list.selectItems('Select Users',
    'sys_user',
    'active=true',
    selectedItems,
    null
  ).then(function(selections) {
    if (selections) {
      console.log('Selected items in list.', selections);
    } else {
      console.log('User cancelled selecting items.');
    }
  }, function(error) {
    console.log(error);
  });
}
```
cabrillo.modal

Cabrillo JS functions for presenting web content inside of native modals.

cabrillo.modal - dismissModal( Object results)

Use to dismiss a modal that has been presented with the presentModal() function.

A presented modal is responsible for dismissing itself and passing any results back to the presenting context. The dismissModal() function must be called from the presented context not from the presenting context.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>results</td>
<td>Object</td>
<td>An object to pass back to the presenting context when the presented context dismisses itself. Optional.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>promise</td>
<td>If successful, an undefined object, otherwise an error.</td>
</tr>
</tbody>
</table>

```javascript
// Any object can be passed back to the presenting context when the presented context dismisses itself.
var results = {
    team: 'Mobile'
    company: 'ServiceNow'
}

cabrillo.modal.dismissModal(results).then(function() {
    console.log('Modal was dismissed and results were passed to presenting context.');
}, function(error) {
    console.log(error);
});
```

cabrillo.modal - presentModal( String title, String url, String closeButtonStyle, String modalPresentationStyle)

Presents content in a native modal interface.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td>Title of the modal interface.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>url</td>
<td>String</td>
<td>The URL to open the modal. This must be an internal instance URL (fully qualified or relative; a relative URL is preferred).</td>
</tr>
<tr>
<td>closeButtonStyle</td>
<td>String</td>
<td>Must be one of cabrillo.modal.CLOSE_BUTTON_STYLE_CANCEL, cabrillo.modal.CLOSE_BUTTON_STYLE_CLOSE, or cabrillo.modal.CLOSE_BUTTON_STYLE_DONE.</td>
</tr>
<tr>
<td>modalPresentationStyle</td>
<td>String</td>
<td>Must be cabrillo.modal.MODAL_PRESENTATION_STYLE_FULL_SCREEN or cabrillo.modal.MODAL_PRESENTATION_STYLE_FORM_SHEET.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>promise</td>
<td>If successful, a Cabrillo.ModalResponse object, otherwise an error.</td>
</tr>
</tbody>
</table>

Present a native modal that loads a custom URL. This presents a custom Service Portal page in a form sheet style modal. The promise is fulfilled when the modal is dismissed. See the `dismissModal()` function for custom dismissal capabilities.

```javascript
cabrillo.modal.presentModal('Portal Page', '/$sp.do?id=my_modal_page',
    cabrillo.modal.CLOSE_BUTTON_STYLE_CLOSE,
    cabrillo.modal.MODAL_PRESENTATION_STYLE_FORM_SHEET
).then(function(response) {
    // The results from the modal are in a results key on the response object.
    var results = response && response.results ?
        response.results : null;
    if (results) {
        console.log('Modal dismissed with results.', results);
    } else {
        console.log('Modal dismissed without results.');
    }
}, function(error) {
    console.log(error);
});
```

cabrillo.navigation

Cabrillo JS functions for forward and backward navigation.

cabrillo.navigation - goBack()

Enables backward navigation.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if Cabrillo JS will navigate backward; otherwise false.</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
cabrillo.navigation.goBack();
```

cabrillo.navigation - goto(String url, Cabrillo.NavigationRequest request)

Enables forward navigation.

Use the request parameter not the url parameter for list or record navigation.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>String</td>
<td>An URL to navigate to. This should be used for custom URL navigation. Optional.</td>
</tr>
<tr>
<td>request</td>
<td>Cabrillo.NavigationRequest</td>
<td>Describes the list or record to navigate to. Optional.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if Cabrillo JS will navigate; otherwise returns false.</td>
</tr>
</tbody>
</table>

Navigate to a URL.

```javascript
cabrillo.navigation.goto('/$sp.do?id=my_custom_page');
```

Navigate to a list. The request parameter is preferred over the url parameter for list navigation.

```javascript
// A Cabrillo.NavigationRequest dictionary that specifies a list of active incidents.  
var request = {  
    table: 'incident',
};
```
Navigate to a record. The request parameter is preferred over the url parameter for record navigation.

```javascript
var request = {
    table: 'incident',
    sysId: 'a9e30c7dc61122760116894de7bcc7bd'
};
cabrillo.navigation.goto(null, request);
```

Navigate to a new record.

```javascript
// The new record will be seeded with the encoded query.
var request = {
    table: 'incident',
    sysId: '-1',
    query: 'short_description=This is a new incident.'
};
cabrillo.navigation.goto(null, request);
```

cabrillo.viewLayout

Cabrillo JS functions to provide access to native UI elements like buttons and spinners.

cabrillo.viewLayout - hideSpinner()

Hides a native spinner in the current interface.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
cabrillo.viewLayout.hideSpinner();
```
cabrillo.viewLayout - setBottomButtons( Array buttons, Function handler)

Set buttons at the bottom of the current interface.
Currently supports a maximum of one button in the array. Images and badges are not supported in bottom buttons.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttons</td>
<td>Array of Cabrillo.Button objects</td>
<td>Describes the buttons to set. A maximum of one button is currently supported.</td>
</tr>
<tr>
<td>handler</td>
<td>Function</td>
<td>The function to call when a button is tapped. The function has no return value and takes the selected button index as its only parameter. The function must have a single parameter that is a Number.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>promise</td>
<td>If successful, is undefined; otherwise, an error.</td>
</tr>
</tbody>
</table>

Set bottom buttons

```javascript
var buttons = [
  {
    title: 'Add to Cart',
    enabled: true,
    backgroundColor: '#3091F9',
    textColor: '#FFFFFF'
  }
];
cabrillo.viewLayout.setBottomButtons(buttons,
  function(buttonIndex) {
    console.log('Received an event from the button.');
  }).then(function() {
    console.log('Buttons were set.');
  }, function() {
    console.log('Failed to register buttons.');
});
```

Clear bottom buttons

```javascript
cabrillo.viewLayout.setBottomButtons();
```
**cabrillo.viewLayout - setNavigationBarButtons( Array buttons, Function handler)**

Set buttons in the navigation bar of the current interface.

Images and badges for buttons that appear in the overflow button menu are omitted. For this reason, it’s best to provide a title and an image name when setting an image button in the navigation bar.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttons</td>
<td>Array of Cabrillo.Button objects.</td>
<td>Describes the buttons to set. Buttons may overflow into an additional menu as needed.</td>
</tr>
<tr>
<td>handler</td>
<td>Function</td>
<td>The function to call when a button is tapped. The function has no return value and takes the selected button index as its only parameter. The function must have a single parameter that is a Number.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>promise</td>
<td>If successful is undefined; otherwise an error.</td>
</tr>
</tbody>
</table>

Set navigation bar buttons

```javascript
var buttons = [
    {
        title: 'Save',
        enabled: true
    }
];

cabrillo.viewLayout.setNavigationBarButtons(buttons,
    function(buttonIndex) {
        console.log('Received an event from the button.');
    }).then(function() {
        console.log('Buttons added.');
    }, function() {
        console.log('Failed to register buttons.');
    });
```

Handle multiple buttons

```javascript
var buttons = [
    {
        title: 'Save',
        enabled: true
    },
```
{  
   title: 'Delete',  
   enabled: true  
};

cabrillo.viewLayout.setNavigationBarButtons(buttons,  
   function(buttonIndex) {  
      switch (buttonIndex) {  
         case 0:  
            console.log('Received an event from the Save button.');  
            break;  
         case 1:  
            console.log('Received an event from the Delete button.');  
            break;  
      }  
   }).then(function() {  
      console.log('Buttons were set.');  
   }, function() {  
      console.log('Failed to register buttons.');  
   });

Buttons placed in the navigation bar can be represented by an image.

var buttons = [  
   {  
      title: 'Compose',  
      imageName: 'compose',  
      enabled: true  
   }  
];

cabrillo.viewLayout.setNavigationBarButtons(buttons,  
   function(buttonIndex) {  
      console.log('Received an event from the button.');  
   }).then(function() {  
      console.log('Buttons were set.');  
   }, function() {  
      console.log('Failed to register buttons.');  
   });

Buttons are placed in the overflow button menu as needed. To force a button into the overflow button menu, set the button's buttonStyle property.

var buttons = [  
   {  
      title: 'Save',  
      buttonStyle: cabrillo.viewLayout.MORE_MENU_BUTTON_STYLE,  
      enabled: true  
   }  
];

cabrillo.viewLayout.setNavigationBarButtons(buttons,  
   function(buttonIndex) {  
      console.log('Received an event from the button.');  
   }).then(function() {  
      // Your code here  
   });
console.log('Buttons were set.');
}, function() {
    console.log('Failed to register buttons.');
});

Buttons may have a badge when placed in the navigation bar. This example sets a button with a shopping cart icon and a badge count of 3. The badge has a blue background with white text.

```javascript
var buttons = [
  {
    title: 'Cart',
    imageName: 'cart',
    badgeCount: 3,
    backgroundColor: '#3091F9',
    textColor: '#FFFFFF',
    enabled: true
  }
];
cabrillo.viewLayout.setNavigationBarButtons(buttons,
  function(buttonIndex) {
    console.log('Received an event from the button.');
  }).then(function() {
    console.log('Buttons were set.');
  }, function() {
    console.log('Failed to register buttons.');
});
```

To clear navigation bar buttons.

```javascript
cabrillo.viewLayout.setNavigationBarButtons();
```

cabrillo.viewLayout - setTitle( String title)

Set the current interface title.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td>The title of the interface.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

cabrillo.viewLayout.setTitle('My Title');
cabrillo.viewLayout - showSpinner()

Shows a native spinner in the current interface.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
cabrillo.viewLayout.showSpinner();
```

CatalogJS - Scoped

CatalogJS API enables you to use methods to check and retrieve catalog-specific properties. To use this class in a scoped application, use the sn_sc namespace identifier. The Service Catalog Scoped API plugin (com.glideapp.servicecatalog.scoped.api) that is enabled by default is required to access the CatalogJS API.

CatalogJS - Catalog(GlideRecord gr)

Creates an instance of the catalog class for the specified glide record object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>Object</td>
<td>Glide Record pointing to the sc_catalog table.</td>
</tr>
</tbody>
</table>

Example:

```
var gr = new GlideRecord('sc_catalog');
gr.addQuery('sys_id','e0d08b13c3330100c8b837659bba8fb4');
gr.query();
var Catalog = new sn_sc.Catalog(gr);
```

CatalogJS - Catalog(String sys_id)

Creates an instance of the Catalog class with the specified sys_id.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>sys_id of the Catalog.</td>
</tr>
</tbody>
</table>

Example:

```javascript
new sn_sc.Catalog(catalog_sys_id);
var Catalog = new
sn_sc.Catalog("31bea3d53790200044e0bfc8bcbe5dec");
```

CatalogJS - canView(boolean mobile)

Specifies if the catalog is viewable for the user.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mobile</td>
<td>Boolean</td>
<td>True if the view is mobile view. Else, false.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the catalog is viewable for the user.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog=new
sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log (catalog.canView(true));
```

Output:

```
true
```

CatalogJS - getAvailableCatalog()

If only one active catalog exists, that catalog is returned. Else, the earliest catalog created is returned, from the list of the catalogs that the user can view. If no catalog is available, null is returned.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Object pointing to the earliest catalog that the user can view.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog = sn_sc.Catalog.getAvailableCatalog()
```

Output:

```
catalog is the CatalogJS object
```

---

**CatalogJS - getBackgroundColor()**

Returns the catalog background color.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Background color of the catalog.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog=new
sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log(catalog.getBackgroundColor());
```

Output:

```
#FFFFFF
```

---

**CatalogJS - getCatalogCount()**

Specifies the number of catalogs active in the catalog table.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Number of catalogs available in the catalog table.</td>
</tr>
</tbody>
</table>

Example:

```javascript
console.log (sn_sc.Catalog.getCatalogCount());
```

Output:

```
3
```

**CatalogJS - getCategories()**

Returns the categories for the specified catalog.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>Returns the categories for the specified catalog.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog=new sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log(catalog.getCategories());
```

Output:

```
  0:
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
CatalogJS - getCategoryIds()

Specifies the sys_ids of the categories in the specified catalog.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>Returns the sys_ids of the categories in the specified catalog.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog=new sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log(catalog.getCategoryIds());
```

Output:

```
0:"e15706fc0a0a0aa7007fc21e1ab70c2f"
  1 :
    "95fc11615f1211001c9b2572f2b477c6"
    2 :
      "900682363731300054b6a3549dbe5d5f"
      3 :
        "d2f716fcc611227a015a142fa0b262c1"
```
CatalogJS - getDescription()

Specifies the catalog description.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Catalog description.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog=new sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log(catalog.getDescription());
```

Output:

Service Catalog - IT Now

CatalogJS - getDesktopImageSRC()

Returns the catalog desktop image value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Catalog desktop image value.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog=new sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log(catalog.getDesktopImageSRC());
```
CatalogJS - getGr()

Returns the catalog gliderecord.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>GlideRecord of the catalog.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog=new sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
data.history=catalog.getGr();
```

Output:

```javascript
{}
```

```
  background_color
  :{}

description
  :
{}

desktop_continue_shopping
  :
{}

desktop_home_page
  :
{}

desktop_image
  :
{}
editors
:
()

enable_wish_list
:
()

manager
:
()

sys_class_name
:
()

sys_created_by
:
()

sys_created_on
:
()

sys_id
:
()

sys_meta
:
{calculation: "", matchable: "1", dynamic_ref_qual: "", choice_field: "", multi_text: "",

sys_mod_count
:
()

sys_name
:
()

sys_package
:
CatalogJS - `getHeaderIconSRC()`

Returns the catalog header icon.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Catalog header icon.</td>
</tr>
</tbody>
</table>
Example:

```javascript
var catalog=new sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log(catalog.getHeaderIconSRC());
```

Output:

```
null
```

**CatalogJS - getID()**

Specifies the sys_id of the catalog.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>sys_id of the catalog.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog=new sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log(catalog.getId());
```

Output:

```
e0d08b13c3330100c8b837659bba8fb4
```

**CatalogJS - getTitle()**

Returns the title of the catalog.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Title of the catalog.</td>
</tr>
</tbody>
</table>
Example:

```javascript
var catalog = new sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log(catalog.getTitle());
```

Output:

```
Service Catalog
```

**CatalogJS - hasCategories()**

Specifies if the catalog has categories.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the catalog has categories. Else returns false.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog = new sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log(catalog.hasCategories());
```

Output:

```
false
```

**CatalogJS - hasItems()**

Specifies if the catalog has catalog items.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the catalog has catalog items. Else returns false.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog=new sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log(catalog.hasItems());
```

Output:

```
true
```

CatalogJS - isWishlistEnabled()

Specifies if the wish list is enabled for a catalog.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if wish list is enabled for a catalog. Else returns false.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var catalog=new sn_sc.Catalog("e0d08b13c3330100c8b837659bba8fb4");
console.log(catalog.isWishlistEnabled());
```

Output:

```
true
```

CatItem

CatItem API enables you to create and modify service catalog items using scripts.

CatItem - canViewOnSearch(boolean isMobile)

Specifies if the user has access to view the catalog item on global search.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isMobile</td>
<td>Boolean</td>
<td>True if the search is in mobile view. Else, false.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the user has access to view the catalog item on global search. Else, returns false.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var cart=new sn_sc.CatItem("04b7e94b4f7b4200086eed18110c7fd");
data.history=cart.canViewOnSearch('false');
```

Output:

```
false
```

CatItem - getFirstAccessibleCategoryForSearch(String catalogId)

Specifies the first category that the user can view in a catalog.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalogId</td>
<td>String</td>
<td>sys_id of the catalog.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>sys_id of the first category that the user can view in a catalog.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var CatItem=new sn_sc.CatItem("04b7e94b4f7b4200086eed18110c7fd");
console.log(CatItem.getFirstAccessibleCategoryForSearch("e0d08b13c3330100c8b837659bba8fb4"));
```

Output:

```
d258b953c611227a0146101fb1be7c31
```
CatItem - getRecordClass()

Returns the class name for the current catalog item record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Class name for the current catalog item record.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var CatItem=new sn_sc.CatItem("04b7e94b4f7b4200086eed18110c7fd");
console.log(CatItem.getRecordClass());
```

Output:

```
sc_cat_item
```

CatItem - isVisibleServicePortal()

Specifies if the catalog item is available in service portal.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the catalog item is available in service portal. Else, returns false.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var CatItem=new sn_sc.CatItem("04b7e94b4f7b4200086eed18110c7fd");
data.history=CatItem.isVisibleServicePortal();
```
Output:

true

Scoped CatItem - availableForUserCriteria(String action, Array[] criteriaIDs)

Adds the Available For user criteria to a catalog item.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>string</td>
<td>Specify add to add the user criteria to the Available For list. Specify delete to delete the user criteria from the Available For list.</td>
</tr>
<tr>
<td>criteriaIDs</td>
<td>Array</td>
<td>Array of the user criteria sys_ids.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var item = new sn_sc.CatItem("31bea3d53790200044e0bfc8bcbe5dec");
item. availableForUserCriteria("add", ["0c441abbc6112275000025157c651c89"]);`n```

Scoped CatItem - notAvailableForUserCriteria(String action, Array[] criteriaIDs)

Adds the Not Available For user criteria to a catalog item.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>String</td>
<td>Specify add to add the user criteria to the Not Available For list. Specify delete to delete the user criteria from the Not Available For list.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var item = new sn_sc.CatItem("31bea3d53790200044e0bfc8bcbe5dec");
item.notAvailableForUserCriteria("add", ["0c441abbc6112275000025157c651c89"]);```

**CatItem - create(Boolean standardUpdate)**

Inserts the defined catalog item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CatItem - deleteRecord(Boolean standardUpdate)**

Deletes the defined catalog item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CatItem - read(Object columns, Boolean standardUpdate)**

Returns a mapping of catalog item attribute values.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
<tr>
<td>columns</td>
<td>Object</td>
<td>Specify the set of columns that you would like the values for.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
</tbody>
</table>

CatItem - setAttributes(Object attributes)

Defines attribute values for this catalog item.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CatItem - setCatalogs(String catalogs)

Defines the catalogs that this catalog item is associated with.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalogs</td>
<td>String</td>
<td>Specify comma-separated list of catalogs that you would like the item to be associated with.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CatItem - setCategories(String categories)

Defines the categories that this catalog item is associated with.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>categories</td>
<td>String</td>
<td>Specify comma-separated list of categories that you would like the item to be associated with.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### CatItem - setImage(String dbImageSysId, String type)

Sets the image of a catalog item to a database image record.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbImageSysId</td>
<td>String</td>
<td>sys_id of an attachment referencing the db_image.</td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>Type can be picture or an icon.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### CatItem - setTableName(String tableName)

Defines the table name for this catalog item.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Specify the name of the table that extends sc_cat_item.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### CatItem - update(Object columnValues, Boolean standardUpdate)

Updates current catalog item with set values.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnValues</td>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CatalogSearch**

CatalogSearch API enables you to search catalog item. To use this class in a scoped application, use the sn_sc namespace identifier. The Service Catalog Scoped API plugin (ID: com.glideapp.servicecatalogScoped.api) that is enabled by default is required to access the CatalogSearch API.

**Scoped CatalogSearch - CatalogSearch()**

Creates an instance of the CatalogSearch class.

```javascript
var gr = new sn_sc.CatalogSearch();
```

**Scoped CatalogSearch - search (String catalogID, String categoryID, String term, Boolean mobile, Boolean depthSearch)**

Searches a catalog item based on a search term. The search can be narrowed down to a catalog category level.

```javascript
var gr = new sn_sc.CatalogSearch();
var catalogID = "exampleID";
var categoryID = "exampleCategory";
var term = "exampleTerm";
var mobile = true;
var depthSearch = true;
var results = gr.search(catalogID, categoryID, term, mobile, depthSearch);
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mobile</td>
<td>Boolean</td>
<td>If true, only catalog items exposed for mobile are searched.</td>
</tr>
<tr>
<td>depthSearch</td>
<td>Boolean</td>
<td>If true, subcategories are also searched.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>Returns the GlideRecord on sc_cat_item matching the search result.</td>
</tr>
</tbody>
</table>

**Example**

```javascript
var gr = new sn_sc.CatalogSearch().search('', '', 'Apple', false, true);
gr.query()
while(var gr = new sn_sc.CatalogSearch().search('', '', 'Apple', false, true);
gr.query()
while(gr.next()) {
    gs.log(gr.name)
}
```

**Output**

```
Apple iPhone 6s Plus
Apple iPhone 6s
Apple MacBook Pro 15"
Apple Watch
Apple Watch Series 2
Apple iPhone 5
Apple iPad 3
Apple iPhone 4 Cable
Apple iPhone 5 Cable
QuickTime Pro
OS X Yosemite
Apple Thunderbolt to Ethernet Adapter
Microsoft Surface Pro 3
OS X Mavericks
Spigen iPhone 6 Case
MacBook Pro Power Adapter
MacBook Air Power Adapter
Spigen iPhone 5/5s Case
Developer Laptop (Mac)
```

**CartJS**

CartJS API enables you to access the shopping cart for a user. To use this class in a scoped application, use the sn_sc namespace identifier. The Service Catalog Scoped API plugin (ID:
com.glideapp.servicecatalog.scoped.api) that is enabled by default is required to access the CartJS API.

**Scoped CartJS - addToCart(Map request)**

Adds the request for a catalog item to the current cart.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>Map</td>
<td>A JSON object that contains the details of the catalog item to be added to the cart. The structure of the request object is:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'sysparm_id': item_id,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'sysparm_quantity': item_quantity,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'variables':{</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'var_name': 'var_value',</td>
</tr>
<tr>
<td></td>
<td></td>
<td>} }</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· item_id: sys_id of the item to be added to the cart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· item_quantity: Number of items to be added. Default value is 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· var_name: Name of the question.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· var_value: Value of the answer (Not the display value).</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON</td>
<td>Structure of the current cart.</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>'subtotal': value,</td>
</tr>
<tr>
<td></td>
<td>'items':[</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>itemName:'',</td>
</tr>
<tr>
<td></td>
<td>quantity:'',</td>
</tr>
<tr>
<td></td>
<td>price:'',</td>
</tr>
<tr>
<td></td>
<td>recurring_price:''</td>
</tr>
<tr>
<td></td>
<td>} ...]</td>
</tr>
</tbody>
</table>
Example

```javascript
var cart = new sn_sc.CartJS();
var item = {
    'sysparm_id': '0d088372371530000158bbfc8bcbe5d02',
    'sysparm_quantity': '1',
    'variables': {
        'carrier': 'at_and_t_mobility',
        'data_plan': '500MB',
        'duration': 'eighteen_months',
        'color': 'slate',
        'storage': 'sixtyfour'
    }
};
var cartDetails = cart.addToCart(item);
gs.info(cartDetails);
```

Output

```javascript
{
    "cart_id": "35ec9e8947a13200e0ef563dbb9a7109",
    "items": [
        {
            "cart_item_id": "35ec9e8947a13200e0ef563dbb9a710a",
            "catalog_item_id": "0d088372371530000158bbfc8bcbe5d02",
            "item_name": "Apple iPhone 5",
            "localized_price": "$799.99",
            "localized_recurring_price": "$30.00",
            "price": "$799.99",
            "quantity": "1",
            "recurring_frequency": "Monthly",
            "recurring_price": "$29.00"
        }
    ],
    "subtotal": "$799.99"
}
```

**Scoped CartJS - CartJS()**

Creates an instance of the CartJS class for the default cart of the user who is currently logged in.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var cart = new sn_sc.CartJS();
```

**Scoped CartJS - CartJS(String cartName)**

Creates an instance of the CartJS class with the name of a defined cart for the user who is currently logged in.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cartName</td>
<td>String</td>
<td>Name of a defined cart for the user who is currently logged in.</td>
</tr>
</tbody>
</table>

```javascript
var cart = new sn_sc.CartJS(cart1);
```

Scoped CartJS - checkoutCart()

Performs the cart checkout. If the two-step checkout is enabled, returns the order summary. If the two-step checkout is disabled, the cart is submitted and details of the generated request are returned.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>JSON</td>
<td>If the two-step checkout is enabled, the summary of the items in the cart is returned.</td>
<td></td>
</tr>
</tbody>
</table>

```json
{
  "subtotal_price": "",
  "subtotal_recurring_frequency": "",
  "delivery_address": "",
  "special_instructions": "",
  "total_title": "",
  "requested_for_user": "System Administrator",
  "requested_for": "6816f79cc0a8016401c5a33be04be441",
  "daily": [
    "frequency_subtotal": "",
    "items": [{}], ...
  ],
  "monthly": [
    "frequency_subtotal": "",
    "items": [{}], ...
  ],
  "annually": [
    "frequency_subtotal": "",
    "items": [{}], ...
  ],
  "none": [
    "frequency_subtotal": "",
    "items": [{}], ...
  ]
}
```

If the two-step checkout is disabled:

```json
{
  'request_id' : "sys_id of the generated request",
  "request_number": "Number of the generated request"
}
```

Example

```javascript
var cart = new sn_sc.CartJS();
var checkoutInfo = cart.checkoutCart();
gs.info(checkoutInfo);
```

Output

If two step checkout enabled:

```javascript
{"subtotal_price": "$2,748.49", "subtotal_recurring_frequency": "", "delivery_address": "", "special_instructions": "", "total_title": "Total", "requested_for_user": "System Administrator", "requested_for": "6816f79cc0a8016401c5a33be04be441", "weekly": 
{"subtotal_price": "$399.50", "subtotal_recurring_frequency": "Weekly", "subtotal_recurring_price": "$0.00", "total_title": "Total", "items": [{"catalog_item_id": "e90a0f7237153000158bbfc8bcb6e5d7e", "variables": {}}], "quantity": 1, "localized_price": "$399.50", "price": "$399.50", "recurring_frequency": "Weekly", "recurring_price": "$0.00", "item_name": "Brother Network-Ready Color Laser Printer", "cart_item_id": "f3f5c9b3c322320076173b0ac3d3ae00", "delivery_time": "2 Days"]}, "show_subtotal_price": "true", "subtotal_title": "Subtotal"}, "show_subtotal_recurring_price": "true", "monthly": 
{"subtotal_price": "$849.99", "subtotal_recurring_frequency": "Monthly", "subtotal_recurring_price": "$31.00", "total_title": "Total", "items": [{"catalog_item_id": "d82ea08510247200964f77f7e6c4e", "variables": {}}], "quantity": 1, "localized_price": "$849.99", "price": "$849.99", "recurring_frequency": "Monthly", "recurring_price": "$31.00", "item_name": "Brother Network-Ready Color Laser Printer", "cart_item_id": "f3f5c9b3c322320076173b0ac3d3ae00", "delivery_time": "2 Days"]}
```
Scoped CartJS - empty()

Deletes the current cart.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var cart = new sn_sc.CartJS();
cart.empty();
```

Scoped CartJS - getCartID()

Returns the cart id of the current cart.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>sys_id for the current cart.</td>
</tr>
</tbody>
</table>
Example

```javascript
var cart = new sn_sc.CartJS();
cart.getCartID();
gs.info(cartId);
```

Output

039c516237b1300054b6a3549dbe5dfc

**Scoped CartJS - getCartItems()**

Returns the GlideRecord for the cart item (sc_cart_item) in the current cart.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>GlideRecord pointing to cart items in the current cart.</td>
</tr>
</tbody>
</table>

Example

```javascript
var cart = new sn_sc.CartJS();
cart.getCartItems();
gs.info(CartItems)
```

Output

Apple iPhone 6s Plus  
Apple iPhone 6s  
Apple MacBook Pro 15"  

**Scoped CartJS - getDeliveryAddress()**

Gets the delivery address for the current cart.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Delivery address for the current cart.</td>
</tr>
</tbody>
</table>

**Example**

```javascript
var cart = new sn_sc.CartJS();
cart.setDeliveryAddress("Brasilia, Brasil");
cart.getDeliveryAddress();
gs.info(DeliveryAddress);
```

**Output**

Brasilia, Brasil

**Scoped CartJS - getRequestedFor()**

Gets the sys_id from the sys_user record of the user for whom the cart is requested.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>sys_id from the sys_user record of the user for whom the cart is requested.</td>
</tr>
</tbody>
</table>

**Example**

```javascript
var cart = new sn_sc.CartJS();
cart.setRequestedFor("039c516237b1300054b6a3549dbe5dfc");
cart.getRequestedFor();
gs.info(cartId);
```

**Output**

039c516237b1300054b6a3549dbe5dfc

**Scoped CartJS - getRequestedForDisplayName()**

Gets the name from the user record of the user for whom the current cart is requested.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Name from the user record of the user for whom the current cart is requested.</td>
</tr>
</tbody>
</table>

Example

```javascript
var cart = new sn_sc.CartJS();
cart.getProjectedCustomerName();
gs.info(ProjectedCustomerName);
```

Output

Abel Tutor

Scoped CartJS - getSpecialInstructions()

Gets the special instructions for the current cart.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Special instructions for the current cart.</td>
</tr>
</tbody>
</table>

Example

```javascript
var cart = new sn_sc.CartJS();
cart.setSpecialInstructions("Delivery before 8 AM.");
cart.getSpecialInstructions();
gs.info(SpecialInstructions);
```

Output

Delivery before 8 AM.
**Scoped CartJS - orderNow(Map request)**

Orders a single item. If two-step checkout is enabled, the item is added to the cart and the cart Sys ID is returned. If two-step checkout is disabled, the item is ordered and the generated request Sys ID and number is returned.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>Map</td>
<td>JSON object that contains details of the catalog item to order.</td>
</tr>
<tr>
<td>request.delivery_address</td>
<td>String</td>
<td>Address to which to deliver the items. Default: Address of user</td>
</tr>
<tr>
<td>request.sysparm_id</td>
<td>String</td>
<td>Required. The sys_id of the item to purchase.</td>
</tr>
<tr>
<td>request.special_instructions</td>
<td>String</td>
<td>Instructions to follow when processing the order.</td>
</tr>
<tr>
<td>request.sysparm_quantity</td>
<td>Number</td>
<td>Quantity of the specified item to purchase. Default: 1</td>
</tr>
<tr>
<td>request.sysparm_requested_for</td>
<td>String</td>
<td>The sys_id of the user for whom the item is requested. Default: Session user</td>
</tr>
<tr>
<td>request.variables</td>
<td>Array</td>
<td>Questions and customer answers associated with the item.</td>
</tr>
<tr>
<td>request.variables.var_name</td>
<td>String</td>
<td>Name of the question.</td>
</tr>
<tr>
<td>request.variables.var_value</td>
<td>String</td>
<td>Customer's response to the associated question.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON key/value pairs</td>
<td>Output if two-step checkout is enabled:</td>
</tr>
</tbody>
</table>
|                             | {
|                             |   'cart_id': '<sys_id of the cart to which the items were added>'           |
|                             | }                                                                           |
|                             | Output if two-step checkout is disabled:                                   |
|                             | {
|                             |   'request_id': '<sys_id of the generated request>',                       |
|                             |   'request_number': '<Number of the generated request>'                    |
|                             | }                                                                           |

Example

```javascript
var cart = new sn_sc.CartJS();
var request = {
  'sysparm_id': '0d08837237153000158bbfc8b5e502',
  'sysparm_quantity': '1',
  'variables':{
    'carrier': 'at_and_t_mobility',
    'data_plan': '500MB',
    'duration': 'eighteen_months',
    'color': 'slate',
    'storage': 'sixtyfour'
  }
}
var cartDetails = cart.orderNow(request);
gs.info(cartDetails);
```

Output

```javascript
// If two-step checkout is enabled:
{"cart_id":"55384df3c322320076173b0ac3d3aec5"}

// If two-step checkout is disabled:
{"request_id":"4c690137c322320076173b0ac3d3ae03",
"request_number": "REQ0010003"}
```

Scoped CartJS - setDeliveryAddress(String address)

Sets the delivery address for the current cart.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>String</td>
<td>Delivery address for the current cart.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var cart = new sn_sc.CartJS();
cart.setDeliveryAddress("Brasilia, Brasil");
```

**Scoped CartJS - setSpecialInstructions(String specialInstructions)**

Sets the special instructions for the current cart.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>specialInstructions</td>
<td>String</td>
<td>Special instructions for the current cart.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var cart = new sn_sc.CartJS();
cart.setSpecialInstructions("Delivery before 8 AM.");
```

**Scoped CartJS - submitOrder(Map request)**

Updates special instructions, requested for, and delivery address from the `request` parameter and performs the cart checkout. Use this API to modify the mentioned parameters of the cart and perform the cart checkout simultaneously. Missing parameters in the `request` object will have their default value.
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>Map</td>
<td>A JSON object that contains details of the cart to be submitted. The structure of the request object is:</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'special_instructions': 'instructions',</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'sysparm_requested_for': requested_for,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'delivery_address': 'address'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· instructions: Special instructions for the request.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· requested_for: sys_id of the requested_for user.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· address: Delivery address for the request.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON</td>
<td>Structure of the cart.</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>'request_id': 'sys_id of the generated Request',</td>
</tr>
<tr>
<td></td>
<td>'request_number': 'Number of the generated Request'</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

## Example

```javascript
var cart = new sn_sc.CartJS();
var request = {
  'special_instructions': 'Delivery only in working hours',
  'requested_for': '62826bf03710200044f84d0f7b6e3d1b',
  'delivery_address': 'Brasilia, Brasil',
};
var requestDetails = cart.submitOrder(request);
gs.info(requestDetails);
```
Scoped CartJS - updateItem(Map request, String cart_item_id)

Updates an item in the cart.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>Map</td>
<td>A JSON object that contains details of the catalog item to be updated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The structure of the request object is:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>`{</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'sysparm_quantity': item_quantity,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'sysparm_requested_for': requested_for,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'variables': {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'var_name':</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'var_value',</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- item_quantity: Number of items to be added. Default value is 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- var_name: Name of the question.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- var_value: Value of the answer (Not the display value).</td>
</tr>
<tr>
<td>cart_item_id</td>
<td>String</td>
<td>sys_id of the cart item to be modified.</td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON</td>
<td>Details of the cart.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Example

```javascript
var cart = new sn_sc.CartJS();
var request =
{
    'sysparm_quantity': '1',
    'variables':{
        'carrier': 'at_and_t_mobility',
        'data_plan': '500MB',
        'duration': 'eighteen_months',
        'color': 'slate',
        'storage': 'sixtyfour'
    }
};
var cart_item_id = "4d69b672c322320076173b0ac3d3ae79";
var cartDetails = cart.updateItem(request, cart_item_id);
gs.info(cartDetails);```

### Output

```javascript
{ "cart_id":"35ec9e8947a13200e0ef563dbb9a7109", "items": [ { "cart_item_id":"35ec9e8947a13200e0ef563dbb9a710a", "catalog_item_id":"0d08837237153000158bdfc8bcb5d02", "item_name":"Apple iPhone 5", "localized_price":"$799.99", "localized_recurring_price":"$30.00", "price":"$799.99", "quantity":1, "recurring_frequency":"Monthly", "recurring_price":"$29.00" } ] "subtotal":"$799.99" }
```

### CartJS - canViewRF()

Specifies if the current user has the required role to edit the Request for field.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the user has the required role to edit the requested for field.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var cart=new sn_sc.CartJS();
console.log(cart.canViewRF());
```

Output:

`true`

CartJS - getCartDetails()

Returns the cart details.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Object pointing to the current cart details.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var cart=new sn_sc.CartJS();
console.log (cart.getCartDetails());
```

Output:

```javascript
delivery_address :
  "Brasilia,Brasil"
  name :
    "DEFAULT"
  requested_for :
    "62826bf03710200044e0bfc8bcbe5df1"
  requested_for_display_name :
    "Abel Tuter"
```
CartJS - setRequestedFor(String user)

Sets the sys_id in the sys_user record of the user for whom the cart is requested.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>String</td>
<td>sys_id to be set in the sys_user record of the user for whom the cart is requested.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var cart = new sn_sc.CartJS();
cart.setRequestedFor("039c516237b1300054b6a3549dbe5dfc")
```

CatalogItemVariable

CatalogItemVariable API enables you to create and modify service catalog item variables using scripts.

CatalogItemVariable - create(Boolean standardUpdate)

Insert the defined catalog item variable.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Return the sys_id of the inserted variable record.</td>
</tr>
</tbody>
</table>

CatalogItemVariable - deleteRecord(Boolean standardUpdate)

Deletes the defined catalog item variable.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CatalogItemVariable - read(Object columns, Boolean standardUpdate)

Returns a mapping of catalog item variable attribute values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columns</td>
<td>Object</td>
<td>Specify the set of columns that you would like the values for.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
</tbody>
</table>

CatalogItemVariable - setAttributes(Object attributes)

Defines attribute values for the specified catalog item variable.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
</tbody>
</table>
CatalogItemVariable - update(Object columnValues, Boolean standardUpdate)

Updates the current catalog item variable with set values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnValues</td>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CatalogClientScript

CatalogClientScript API enables you to create, modify, or delete catalog client script records. To use this class in a scoped application, use the sn_sc namespace identifier. The Service Catalog Scoped API plugin (ID: com.glideapp.servicecatalog.scoped.api) that is enabled by default is required to access the CatalogClientScript API.

Scoped CatalogClientScript - addScript(String script)

Adds a script to the catalog client script.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>script</td>
<td>String</td>
<td>Script to be added to the catalog client script.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```
var catalogClientScript = new sn_sc.CatalogClientScript();
```
catalogClientScript.addScript("function onLoad(){Enter the script} ");

**Scoped CatalogClientScript - appliesToCatalogItem(Boolean flag)**

Specifies if the catalog client script runs on a catalog item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flag</td>
<td>Boolean</td>
<td>If true, the catalog client script runs on the catalog item. If false, the catalog client script does not run on the catalog item.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```javascript
var catalogClientScript = new sn_sc.CatalogClientScript();
catalogClientScript.appliesToCatalogItem(true);
```

**Scoped CatalogClientScript - appliesToCatalogTask(Boolean flag)**

Specifies if the catalog client script runs on a catalog task.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flag</td>
<td>Boolean</td>
<td>If true, the catalog client script runs on the catalog task. If false, the catalog client script does not run on the catalog task.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```javascript
var catalogClientScript = new sn_sc.CatalogClientScript();
catalogClientScript.appliesToCatalogTask(true);
```
Scoped CatalogClientScript - appliesToRequestedItem(Boolean flag)

Specifies if the catalog client script runs on a requested item.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flag</td>
<td>Boolean</td>
<td>If true, the catalog client script runs on the requested item. If false, the catalog client script does not run on the requested item.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```python
var catalogClientScript = new sn_sc.CatalogClientScript();
catalogClientScript.appliesToRequestedItem(true);
```

Scoped CatalogClientScript - appliesToTargetRecord(Boolean flag)

Specifies if the catalog client script runs on a requested item.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flag</td>
<td>Boolean</td>
<td>If true, the catalog client script runs on the target record. If false, the catalog client script does not run on the target record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```python
var catalogClientScript = new sn_sc.CatalogClientScript();
catalogClientScript.appliesToTargetRecord(true);
```
**Scoped CatalogClientScript - CatalogClientScript()**

Creates an instance of the CatalogClientScript class.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
var catalogClientScript = new sn_sc.CatalogClientScript();
```

**Scoped CatalogClientScript - create(Boolean standardUpdate)**

Inserts the defined catalog client script in the catalog_script_client table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>sys_id of the catalog client script.</td>
</tr>
</tbody>
</table>

Example

```
var catalogClientScript = new sn_sc.CatalogClientScript();
catalogClientScript.setAttributes({"name": "My Catalog Item", "applies_to": "item", "ui_type": "desktop", "type": "onLoad"});
catalogClientScript.appliesToCatalogItem(true);
catalogClientScript.appliesToRequestedItem(true);
catalogClientScript.appliesToCatalogTask(true);
catalogClientScript.appliesToTargetRecord(true);
var catalogClientScriptId = catalogClientScript.create();
gs.info(catalogClientScriptId);
```

Output

```
039c516237b1300054b6a3549dbe5dfe
```

**Scoped CatalogClientScript - deleteRecord(String sys_id, Boolean standardUpdate)**

Deletes the defined catalog client script.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>sys_id of the catalog client script.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example:

```javascript
var sys_id = "039c516237b1300054b6a3549dbe5dfc";
var catalogClientScript = new sn_sc.CatalogClientScript();
catalogClientScript.deleteRecord("039c516237b1300054b6a3549dbe5dfc");
```

Scoped CatalogClientScript - setAttributes(Map attributes)

Defines attribute values for the catalog client script.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Map</td>
<td>A JSON object that has mapping for the field and value pairs.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var catalogClientScript = new sn_sc.CatalogClientScript();
catalogClientScript.setAttributes({"name": "My Catalog Item", "applies_to": "catalog_item", "ui_type": "desktop", "type": "onLoad"});
```

Scoped CatalogClientScript - setCatalogItem(String sys_id)

Associates a catalog item with the catalog client script.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>sys_id of the catalog item.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### Example

```javascript
var catalogClientScript = new sn_sc.CatalogClientScript();
catalogClientScript.setCatalogItem("039c516237b1300054b6a3549dbe5dfc");
```

**Scoped CatalogClientScript - setOnChangeVariable(String sys_id)**

Runs the catalog client script when a variable value is updated.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>sys_id of the variable.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### Example

```javascript
var catalogClientScript = new sn_sc.CatalogClientScript();
catalogClientScript.setOnChangeVariable("039c516237b1300054b6a3549dbe5dfc");
```

**Scoped CatalogClientScript - setVariableSet(String sys_id)**

Associates a variable set with the catalog client script.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>sys_id of the variable set.</td>
</tr>
</tbody>
</table>
CatalogItemVariableSet

CatalogItemVariableSet API enables you to create and modify service catalog item variable sets using scripts.

CatalogItemVariableSet - create(Boolean standardUpdate)

Inserts the defined catalog item variable set.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CatalogItemVariableSet - deleteRecord(Boolean standardUpdate)

Deletes the defined catalog item variable.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CatalogItemVariableSet - read(Object columns, Boolean standardUpdate)

Returns a mapping of catalog item variable set attribute values.
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columns</td>
<td>Object</td>
<td>Specify the set of columns that you would like the values for.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
</tbody>
</table>

### CatalogItemVariableSet - setAttributes(Object attributes)

Defines attribute values for this catalog item variable set.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
</tbody>
</table>

### CatalogItemVariableSet - update(Object columnValues, Boolean standardUpdate)

Updates the current catalog item variable set with set values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnValues</td>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

### CatalogItemVariableSet - update(Object columnValues, Boolean standardUpdate)

Updates the current catalog item variable set with set values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnValues</td>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

### CatalogItemVariableSet - update(Object columnValues, Boolean standardUpdate)

Updates the current catalog item variable set with set values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnValues</td>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>
CatalogItemVariableSetM2M

CatalogItemVariableSetM2M API enables you to create and modify service catalog item variable set many-to-many (M2Ms) using scripts.

CatalogItemVariableSetM2M - create(Boolean standardUpdate)

Inserts the defined catalog item variable set M2M.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Return the sys_id of the inserted variable record.</td>
</tr>
</tbody>
</table>

CatalogItemVariableSetM2M - deleteRecord(Boolean standardUpdate)

Deletes the defined catalog item variable set M2M.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CatalogItemVariableSetM2M - read(Object columns, Boolean standardUpdate)

Returns a mapping of catalog item variable set M2M attribute values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columns</td>
<td>Object</td>
<td>Specify the set of columns that you would like the values for.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>Object</td>
<td>An object mapping column names to values.</td>
<td></td>
</tr>
</tbody>
</table>

**CatalogItemVariableSetM2M - setAttributes(Object attributes)**

Defines attribute values for this catalog item variable set M2M.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CatalogItemVariableSetM2M - update(Object columnValues, Boolean standardUpdate)**

Updates the current catalog item variable set M2M with set values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnValues</td>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CatCategory**

CatCategory API enables you to create and modify service catalog categories using scripts.

**Scoped CatCategory - availableForUserCriteria(String action, Array[] criterialDs)**

Adds the Available For user criteria to a catalog category.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>String</td>
<td>Specify add to add the user criteria to the Available For list. Specify delete to delete the user criteria from the Available For list.</td>
</tr>
<tr>
<td>criteriaIDs</td>
<td>Array</td>
<td>Array of the user criteria sys_ids.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
var item = new 
sn_sc.CatCategory("31bea3d53790200044e0bfc8bcbe5dec");
item. availableForUserCriteria("add",
["0c441abbc6112275000025157c651c89"]);
```

Scoped CatCategory - notAvailableForUserCriteria(String action, Array() criteriaIDs)

Adds the Not Available For user criteria to a catalog category.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>String</td>
<td>Specify add to add the user criteria to the Not Available For list. Specify delete to delete the user criteria from the Not Available For list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
var item = new 
sn_sc.CatCategory("31bea3d53790200044e0bfc8bcbe5dec");
```
CatCategory - canView(Boolean isMobile, String userSysId)

Determines whether a specified user can view a specified category on a mobile device or desktop.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isMobile</td>
<td>Boolean</td>
<td>Flag that indicates whether to verify the user for access on a mobile device or desktop.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true: validate for mobile.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false: validate for desktop.</td>
</tr>
<tr>
<td>userSysId</td>
<td>String</td>
<td>Unique identifier of the user to validate.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Flag that indicates whether the user can view the associated category.</td>
</tr>
<tr>
<td></td>
<td>• true: user can view the category.</td>
</tr>
<tr>
<td></td>
<td>• false: user cannot view the category.</td>
</tr>
</tbody>
</table>

```javascript
var scopedCategoryObj = new sn_sc.CatCategory('7b1262b9530033007444ddeeff7b12ae');
scopedCategoryObj.canView(false, '62826bf03710200044e0bfc8bcbe5df1');
```

CatCategory - create(Boolean standardUpdate)

Inserts the defined category.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Return the sys_id of the inserted variable record.</td>
</tr>
</tbody>
</table>

```javascript
var categoryCreate = new sn_sc.CatCategory();
categoryCreate.setAttributes({"title" : "test a scoped category", "sc_catalog" : "e0d08b13c3330100c8b837659bba8fb4"});
var categorySysId = categoryCreate.create();
var isValidSysId = categorySysId.match(/^[0-9a-fA-F]{32}$/) == null ? false : true;
global.Assert.assertEquals(true, isValidSysId,"CategorySysId: [+ categorySysId +"] is not valid", true, isValidSysId);
```

**CatCategory - deleteRecord(Boolean standardUpdate)**

Deletes the category record on which the CatCategory class was initially instantiated.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var categoryDelete = new sn_sc.CatCategory(categorySysId);
categoryDelete.deleteRecord();
var category = new sn_sc.CatCategory(categorySysId);
values = category.read({"title" : "", "sc_catalog":""}, false);
global.Assert.assertEquals("", values.title,"Category should title");
```

**CatCategory - read(Object columns, Boolean standardUpdate)**

Returns a mapping of the category.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columns</td>
<td>Object</td>
<td>Specify the set of columns that you would like the values for.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
</tbody>
</table>

```javascript
var category = new sn_sc.CatCategory("a96277509f300200b407b89a442e704e");
var values = category.read({"title" : ",", true});
gs.log(values.title);
```

**CatCategory - setAttributes(Object attributes)**

Defines attribute values for this category.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Object</td>
<td>Set the attributes for new field and value pairs.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var categoryCreate = new sn_sc.CatCategory();
categoryCreate.setAttributes({"title" : "test a scoped category", "sc_catalog" : "e0d08b13c3330100c8b837659bba8fb4");
var categorySysId = categoryCreate.create();
var isValidSysId = categorySysId.match(/^[0-9a-fA-F]{32}$/) == null ? false : true;
global.Assert.assertEquals(true, isValidSysId,"CategorySysId: ["+ categorySysId +"] is not valid", true, isValidSysId);
```

**CatCategory - setTableName(String tableName)**

Define the table name for this category.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Specify the name of the table that extends sc_category.</td>
</tr>
</tbody>
</table>
CatCategory - update(Object columnValues, Boolean standardUpdate)

Use to update current category.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnValues</td>
<td>Object</td>
<td>An object mapping column names to values.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CatCategory - getID()

Returns the sys_id of the category.
### CertificateEncryption

APIS available for encrypting certificates in scoped applications.

Use these methods to generate a hash for the certificate, sign data using a private key, and generate a message authentication code.

**CertificateEncryption - CertificateEncryption()**

Instantiates a CertificateEncryption object in a scoped application.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CertificateEncryption - generateMac(String key, String algorithm, String data)**

Generates the Message Authentication Code (MAC), which is used to authenticate a message.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>String</td>
<td>Key to use to sign the message. Must be Base64 encoded.</td>
</tr>
<tr>
<td>algorithm</td>
<td>String</td>
<td>Algorithm to use to generate the MAC: HmacSHA256, HmacSHA1, HmacMD5, and so on.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>data</td>
<td>String</td>
<td>Data to process.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>MAC in base64 format.</td>
</tr>
</tbody>
</table>

**Example**

```java
var mac = new CertificateEncryption;
var key = "sample_key";
key = GlideSystem.base64Encode(key);
mac.generateMac(key, "HmacSHA256", "sample_data");
```

**CertificateEncryption - getThumbPrint(String certificateID, String algorithm)**

Generates a hash (SHA-1, SHA-256, and so on) for the certificate from Trust Store Cert.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificateID</td>
<td>String</td>
<td>sys_id of the certificate record in the X.509 Certificate (sys_certificate) table.</td>
</tr>
<tr>
<td>algorithm</td>
<td>String</td>
<td>SHA-1, SHA-256, and so on</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Thumbprint in base64 format.</td>
</tr>
</tbody>
</table>

**CertificateEncryption - getThumbPrintFromKeystore(String certificateID, String alias, String algorithm)**

Generates a hash (SHA-1, SHA-256, and so on) for the certificate from the keystore entry.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificateID</td>
<td>String</td>
<td>sys_id of the certificate record in the X.509 Certificate (sys_certificate) table.</td>
</tr>
<tr>
<td>alias</td>
<td>String</td>
<td>Alias name for the certificate.</td>
</tr>
<tr>
<td>algorithm</td>
<td>String</td>
<td>SHA-1, SHA-256, and so on</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Thumbprint in base64 format.</td>
</tr>
</tbody>
</table>

CertificateEncryption - sign(String certificateID, String alias, String aliaspassword, String algorithm, String datatosign)

Signs the data using the private key and the given algorithm.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificateID</td>
<td>String</td>
<td>sys_id of the certificate record in the X.509 Certificate (sys_certificate) table.</td>
</tr>
<tr>
<td>alias</td>
<td>String</td>
<td>Private key name.</td>
</tr>
<tr>
<td>aliaspassword</td>
<td>String</td>
<td>Password for the private key.</td>
</tr>
<tr>
<td>datatosign</td>
<td>String</td>
<td>Data to sign.</td>
</tr>
<tr>
<td>algorithm</td>
<td>String</td>
<td>SHA-1, SHA-256, and so on.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Signed data in base64 format.</td>
</tr>
</tbody>
</table>

```java
var ce = new CertificateEncryption;
ce.sign("recordID", "alias", "password", "SHA-1", "sign this data");
```

CIUtils

CIUtils is a utility class for working with configuration items.

By default, when traversing CI relationships the system will use a max depth of 10. This can be overridden in the glide.relationship.max_depth property.

The maximum number of items returns is 1000. This can be overridden in the glide.relationship.threshold property.

The CIUtils class is available to server-side scripts.

CIUtils - servicesAffectedByCI(String CI_sys_id)

Determines which business services are affected by a specific CI.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI_sys_id</td>
<td>String</td>
<td>The sys_id of a configuration item (cmdb_ci)</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of sys_id values for cmdb_ci records downstream of (or affected by) the specified item.</td>
</tr>
</tbody>
</table>

```javascript
var CIUtil = new CIUtils();

//get a server record
var server = new GlideRecord("cmdb_ci_server");
server.addQuery("name", "lnux100");
server.query();
if (server.next()) {
    //get the affected services, array of ids
    var serviceIds = CIUtil.servicesAffectedByCI(server.getUniqueValue());
    for (var i=0; i < serviceIds.length; i++) {
        //get the service record
        var service = new GlideRecord("cmdb_ci_service");
        service.get(serviceIds[i]);
        gs.print(service.getDisplayValue());
    }
}
```

**Output:**

Client Services
IT Services
Bond Trading

---

**CIUtils - servicesAffectedByTask(GlideRecord task)**

Determines which business services are affected by a task.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>task</td>
<td>GlideRecord</td>
<td>A task GlideRecord, for example incident, change_request, problem</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of sys_id values for CIs downstream of (or affected by) the configuration item referenced by the task's cmdb_ci field.</td>
</tr>
</tbody>
</table>

```javascript
var CIUtil = new CIUtils();

//get an incident record
var inc = new GlideRecord("incident");
inc.addQuery("number", "INC00050");
inc.query();
if (inc.next()) {
    //get the affected services, array of ids
    var serviceIds = CIUtil.servicesAffectedByTask(inc);
    for (var i=0; i < serviceIds.length; i++) {
        //get the service record
        var service = new GlideRecord("cmdb_ci_service");
        service.get(serviceIds[i]);
        gs.print(service.getDisplayValue());
    }
}
```

Output:

IT Services
Email
Windows Mobile
Electronic Messaging
Outlook Web Access (OWA)
Blackberry

ChangeCollisionHelper

Helper functions found in the Change Management Collision Detector Plugin.
Use these methods in server side scripts, or when using AJAX calls on the client. You must have the Change Management Collision Detector Plugin installed to use these methods.

`ChangeCollisionHelper - addCiToChangeAffectedCis(String ci, String changeld)`

Adds the CI to the change's affected CI list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci</td>
<td>String</td>
<td>The sys_id of the configuration item</td>
</tr>
<tr>
<td>changeld</td>
<td>String</td>
<td>The change record's sys_id</td>
</tr>
</tbody>
</table>
### ChangeCollisionHelper - `getAffectedCisByChangeId(String changeld)`

Returns the Affected CI sys_ids for the given change.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>changeld</td>
<td>String</td>
<td>A change record's sys_id</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of sys_ids of affected CIs.</td>
</tr>
</tbody>
</table>

### ChangeCollisionHelper - `getBlackoutsByDate(GlideDateTime startDate, GlideDateTime endDate)`

Returns any blackout that overlap the period defined by startDate and endDate.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startDate</td>
<td>GlideDateTime</td>
<td>The beginning date</td>
</tr>
<tr>
<td>endDate</td>
<td>GlideDateTime</td>
<td>The ending date</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of blackouts (blackoutId:stringSpan).</td>
</tr>
</tbody>
</table>

### ChangeCollisionHelper - `getChangesWithAffectedCi(String ci, GlideDateTime startDate, GlideDateTime endDate)`

Returns changes scheduled in the timespan (defined by startDate and endDate) that have the given CI in their Affected CIs list.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci</td>
<td>String</td>
<td>The configuration item's sys_id</td>
</tr>
<tr>
<td>startDate</td>
<td>GlideDateTime</td>
<td>The beginning date</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of blackouts (blackoutId:stringSpan).</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>endDate</td>
<td>GlideDateTime</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of change record’s sys_ids</td>
</tr>
</tbody>
</table>

**ChangeCollisionHelper - getChangesWithCi(String ci, GlideDateTime startDate, GlideDateTime endDate)**

Returns the changes that are in the timespan (startDate, endDate) and that are linked to the given CI.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci</td>
<td>String</td>
<td>The configuration item’s sys_id</td>
</tr>
<tr>
<td>startDate</td>
<td>GlideDateTime</td>
<td>The beginning date</td>
</tr>
<tr>
<td>endDate</td>
<td>GlideDateTime</td>
<td>The ending date of the time span</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of sys_ids for change records</td>
</tr>
</tbody>
</table>

**ChangeCollisionHelper - getCiMaintenanceSchedule(String ci)**

Returns the Maintenance Schedule for a CI.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci</td>
<td>String</td>
<td>The configuration item’s sys_id</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ChangeCollisionHelper - getDependants(String ci)**

Returns all the CIs that depend on the given CI.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci</td>
<td>String</td>
<td>A configuration item’s sys_id</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of CIs</td>
</tr>
</tbody>
</table>

**ChangeCollisionHelper - getDependencies(String ci)**

Returns all the CIs that the given CI depends on.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci</td>
<td>String</td>
<td>The configuration item’s sys_id</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of CIs</td>
</tr>
</tbody>
</table>

**ChangeCollisionHelper - isCiInAffectedCis(String ci, String changeld)**

Checks if an CI is already in the change's affected CIs list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci</td>
<td>String</td>
<td>The sys_id of the configuration item</td>
</tr>
<tr>
<td>changeld</td>
<td>String</td>
<td>The change record’s sys_id</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the CI already is in the change's affected CI list.</td>
</tr>
</tbody>
</table>

**ChangeCollisionHelper - isDateInCiMaintenanceWindows(GlideDateTime startDate, GlideDateTime endDate, String maintenanceWindow)**

Checks if the time span defined by startDate and endDate falls in the CI’s maintenance window.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startDate</td>
<td>GlideDateTime</td>
<td>The beginning date</td>
</tr>
<tr>
<td>endDate</td>
<td>GlideDateTime</td>
<td>The ending date</td>
</tr>
<tr>
<td>maintenanceWindow</td>
<td>String</td>
<td>The configuration item's sys_id</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the time span is in the CI's maintenance window</td>
</tr>
</tbody>
</table>

**ChangeConflict**

Helper functions found in the Change Management Collision Detector Plugin.

Use these methods in server side scripts, or when using AJAX calls on the client. You must have the Change Management Collision Detector Plugin installed to use these methods.

**ChangeConflict - ChangeConflict(String ciId, String changeld, Number type)**

Creates an instance of ChangeConflict.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciId</td>
<td>String</td>
<td>A configuration item's sys_id</td>
</tr>
<tr>
<td>changeld</td>
<td>String</td>
<td>A change request's sys_id</td>
</tr>
<tr>
<td>type</td>
<td>Number</td>
<td>A value from the conflict's type choice list</td>
</tr>
</tbody>
</table>

**ChangeConflict - toString()**

Returns a string representation of the conflict.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The string representation of the conflict</td>
</tr>
</tbody>
</table>
ChangeConflictHandler

Helper functions found in the Change Management Collision Detector Plugin.

Use these methods in server side scripts, or when using AJAX calls on the client. You must have the Change Management Collision Detector Plugin installed to use these methods.

**ChangeConflictHandler - addChangeConflict(String chgConflict)**

Adds the Change Conflict to a Change Conflict Container.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chgConflict</td>
<td>String</td>
<td>The sys_id of the change conflict</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ChangeConflictHandler - ChangeConflictHander**

Creates an instance of ChangeConflictHandler.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ChangeConflictHandler - deleteConflictsByChangeId(String changeId)**

Deletes conflicts that are associated with the same change request (by sys_id).

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>changeld</td>
<td>String</td>
<td>The sys_id of the change request</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ChangeConflictHandler - getConflicts()**

Returns an array of Change Conflicts from a Change Conflict Container.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of change conflicts</td>
</tr>
</tbody>
</table>

ChangeConflictHandler - saveConflicts()

Writes out the Change Conflicts in a Change Conflict Container array to individual Change Conflict records.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CIData

Utility class for working with CI data structures in JavaScript.

Instances of this class represent a CI (both its base record and any related lists), and the provided methods allow loading from or storing both the base record and related lists.

This class acts as a container for simple classes acting as maps of property name/value pairs. This class contains three kinds of such maps.

- A single map representing the base CI table entry (such as one row cmdb_ci_linux_server and its superclasses). This map can be retrieved with the getData() method.
- Arrays of maps representing the instances of related lists, with each array representing a single related list and each element of such an array representing a single row of that related list. These arrays can be retrieved with the getRelatedList() method.
- Arrays of maps representing the instances of many-to-many tables, with each array representing a single many-to-many table and each element of such an array representing a single row of that many-to-many table. These arrays can be retrieved with the getM2MTable() method.

Use with any server-side discovery script.
CIData - addRelatedList(Array relatedList)

Adds the specified related list to this instance.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>relatedList</td>
<td>Array</td>
<td>The related list</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CIData - CIData(String debugFlag)

Creates an instance of the CIData class.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debugFlag</td>
<td>String</td>
<td>True if debug logging is enabled.</td>
</tr>
</tbody>
</table>

CIData - convertRelatedList(String sensor, String tableName, String refField, String keyName)

Converts the specified related list to a related list in a given sensor.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensor</td>
<td>String</td>
<td>The sensor</td>
</tr>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table name</td>
</tr>
<tr>
<td>refField</td>
<td>String</td>
<td>The reference field</td>
</tr>
<tr>
<td>keyName</td>
<td>String</td>
<td>The key name</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CIData - fromXML(String xml)

Initializes the current CIData instance from the specified XML string.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml</td>
<td>String</td>
<td>An XML string</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### CIData - getData()

Returns the data map in the base CI record.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The data map</td>
</tr>
</tbody>
</table>

#### CIData - getM2MTable(String table, String refField)

Returns an array of data maps in the given many-to-many list (to this CI).

The array is not in any particular order. If this is not a many-to-many list, returns an empty array.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>The name of the table containing the related list. In the case of a many-to-many list, this is the name of the many-to-many table, not the target table.</td>
</tr>
<tr>
<td>refField</td>
<td>String</td>
<td>The name of the field in the related list that refers to this CI. In the case of a many-to-many list, this is the name of the referring field in the many-to-many table, not in the target table.</td>
</tr>
</tbody>
</table>
### CIData - getRelatedList(String table, String refField)

Returns an array of maps of data in the given related list (to this CI).

The array is not in any particular order. In the case of a many-to-many list, this will be an array of instances of the target table, not the many-to-many table. For example, given `cmdb_software_instance` and `installed_on` (a many-to-many table and the field that refers to a CI), this method will return an array of maps representing cmdb_ci_spkg (the target table) instances.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>Name of the table containing the related list. In the case of a many-to-many list, this is the name of the many-to-many table, not the target table.</td>
</tr>
<tr>
<td>refField</td>
<td>String</td>
<td>Name of the field in the related list that refers to this CI. In the case of a many-to-many list, this is the name of the referring field in the many-to-many table, not in the target table.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of maps of data in the specified related list.</td>
</tr>
</tbody>
</table>

### CIData - getRelatedListInstance(String table, String refField)

Returns the instance of CIRelatedList for the given list.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>Name of the table containing the related list. In the case of a many-to-many list, this is the name of the many-to-many table, not the target table.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>refField</td>
<td>String</td>
<td>Name of the field in the related list that refers to this CI. In the case of a many-to-many list, this is the name of the referring field in the many-to-many table, not in the target table.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Instance of the CIRelatedList for the specified list.</td>
</tr>
</tbody>
</table>

**CIData - init()**

Initializes this instance in preparation for loading a new CI.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var cdta = CIData('false');
cdta.init();
```

**CIData - loadFromCI(String cmdb_ci)**

Loads the current data from a specified sys_id.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmdb_ci</td>
<td>String</td>
<td>The sys_id of the CI</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The current data from the specified CI.</td>
</tr>
</tbody>
</table>
```javascript
var cdta = CIData('false');
cdta.init();
cdta.loadFromCI('ccaf9c0a8016400b98a06818d57c7');
```

**CIData - toString()**

Converts the name of this instance to a string.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Name of the instance</td>
</tr>
</tbody>
</table>

**CIData - toXML()**

Returns an XML string containing a serialized version of this instance (including any related lists).

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The XML string containing a serialized version of this instance and any related lists.</td>
</tr>
</tbody>
</table>

```javascript
var cdta = CIData();
cdta.toXML();
```

**CIIdentification**

Main class for discovery CI identification.

Use this with any server-side discovery script for CI identification.

**CIIdentification - CIIdentification(String ciData, String logger)**

Creates an instance of the CIIdentification class.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciData</td>
<td>String</td>
<td>The CI data to identify</td>
</tr>
<tr>
<td>logger</td>
<td>String</td>
<td>The discovery logger</td>
</tr>
</tbody>
</table>

### CIIdentification - debug(String msg)

Logs a message to the CI Identification log if debug logging is turned on.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>The message to log</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### CIIdentification - process()

Identifies the CI. This is the entry point for the entire CI Identification process.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDResult</td>
<td>The result returned by the identifier</td>
</tr>
</tbody>
</table>

### CIIdentifierResult

Gets the result returned by an identifier. Use with any server-side discovery script.

### CIIdentifierResult - CIIdentifierResult(Array matched, Boolean matchable, Boolean terminatedChain)

Creates an instance of the CIIdentifierResult class.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>matched</td>
<td>Array</td>
<td>An array of sys_ids of matching CIs; may have any number of entries including zero.</td>
</tr>
<tr>
<td>matchable</td>
<td>Boolean</td>
<td>True if identifier had sufficient data to match</td>
</tr>
<tr>
<td>terminatedChain</td>
<td>Boolean</td>
<td>True if the identifier chain should stop processing filters. False to continue.</td>
</tr>
</tbody>
</table>

CimCIData

Provides a wrapper for CIM CI data manipulation. Use with any server-side discovery script.

CimCIData - addSerial(String type, String serial)

Adds a serial number to the current CI, by type.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>The serial number type. If out-of-box, use the CimCIData.serialType enum.</td>
</tr>
<tr>
<td>serial</td>
<td>String</td>
<td>The serial number</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current CI</td>
</tr>
</tbody>
</table>

CimCIData - SerialRecord(String type, String serial)

Creates a new serial record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>The serial type to assign to the CI. If out-of-box, uses the CimCIData.serialType enum.</td>
</tr>
<tr>
<td>serial</td>
<td>String</td>
<td>The serial number to assign.</td>
</tr>
</tbody>
</table>
CimClData - setMakeAndModel(String make, String model)
Sets the make and model for the current CI.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>make</td>
<td>String</td>
<td>The manufacturer</td>
</tr>
<tr>
<td>model</td>
<td>String</td>
<td>The model</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CimIDSensor

Performs identification for CIM probe results.
Use in any server-side script to perform identification of CIM probe results.

CimIDSensor - configureTriggeredProbe(String probe)
Configures a triggered probe before it is launched.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>probe</td>
<td>String</td>
<td>An already configured probe that is ready to fire.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True to fire this probe, false to skip it.</td>
</tr>
</tbody>
</table>

CimIDSensor - getInstanceHashToken()
Returns the instance hash token.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The hash token</td>
</tr>
</tbody>
</table>

### CimIDSensor - prepare()

Prepares the probe results.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### CimIDSensor - process(Object results, CimCIData cimData, CimIDSensor sensor)

Processes all CIM probe query results for a CIM ID sensor.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>results</td>
<td>Object</td>
<td>The query results</td>
</tr>
<tr>
<td>cimData</td>
<td>CimCIData</td>
<td>The CI data of the device being identified.</td>
</tr>
<tr>
<td>sensor</td>
<td>CimIDSensor</td>
<td>The wrapping sensor, extends CimIDSensor</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### CimIDSensor - runMultiProbeScript(String script, Object probeResult, String probeRecord)

Runs a multi-probe script against a probe result.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>script</td>
<td>String</td>
<td>The script</td>
</tr>
<tr>
<td>probeResult</td>
<td>Object</td>
<td>The probe result</td>
</tr>
<tr>
<td>probeRecord</td>
<td>String</td>
<td>The probe ID</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CimIDSensor - updateDeviceCount()**

Updates the device count.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CimInstanceToken**

Represents the unique query identity of a CIM instance.

Use with any server-side discovery script.

**CimInstanceToken - getHashToken(String cimomip)**

Parses the instance token.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cimomip</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The hash token</td>
</tr>
</tbody>
</table>
CimInstanceToken - parse(XMLObj instance)

Parses the instance token.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance</td>
<td>XMLObj</td>
<td>The CIM instance</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The parsed instance token, for example, CIM_ClassName(Key1='Value1',Key2='Value2')</td>
</tr>
</tbody>
</table>

CimProbe

Maintains CIM probe meta data.

Use in any server-side script where you need to maintain a CIM probe.

CimProbe - getNamedQueries()

Returns all the named queries for the current CIM probe.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CimProbe - getQueries()

Returns all the queries for the current CIM probe.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of strings of queries</td>
</tr>
</tbody>
</table>
CimProbe - getQueryRecords()

Returns all the records in the CIM Probe table.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The query result</td>
</tr>
</tbody>
</table>

CimProbeResult

Processes CIM probe results.

Use to process CIM probe results in any server-side discovery script.

CimProbeResult - getNamedInstances(String namedQueries)

Returns the named instances for the specified named queries.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>namedQueries</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The named instances</td>
</tr>
</tbody>
</table>

CimProbeResult - getQueries()

Returns all the queries for the current CIM probe.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Array</td>
<td>All Queries</td>
</tr>
</tbody>
</table>
**CIRelatedList**

Utility class for working with CI Related lists.

Each instance of this class contains the details of a single list that is related to a particular CI. The details of this list and the contents of the list are included.

Use with any server-side discovery script.

**CIRelatedList - addRec(String record)**

Adds the specified record to the related list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>String</td>
<td>The record to add</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CIRelatedList - addRecs(String records)**

Sets the given records to the related list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>records</td>
<td>String</td>
<td>The records to set</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CIRelatedList - appendXMLChildFld(String parent, String name, String datum)**

Appends an XML child to a field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>String</td>
<td>The parent element</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The element name</td>
</tr>
<tr>
<td>datum</td>
<td>String</td>
<td>The data to append</td>
</tr>
</tbody>
</table>
CIRelatedList - appendXMLChildRecord(String parent, String name, String record)

Appends an XML child to a record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>String</td>
<td>The parent element</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The element name</td>
</tr>
<tr>
<td>record</td>
<td>String</td>
<td>The record</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CIRelatedList - checkSysCollection()

Checks to see if this is a SysCollection table. If it is, sets the target table name and reference field name.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIRelatedList</td>
<td>An instance of CIRelatedList for the specified list.</td>
</tr>
</tbody>
</table>

CIRelatedList - checkSysM2M()

Checks to see if this is a Sys many-to-many table. If it is, sets the target table name and reference field name.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Boolean</td>
<td>True if it is a sys many-to-many table; otherwise, false.</td>
<td></td>
</tr>
</tbody>
</table>

**CIRelatedList - CIRelatedList(String table, String refField, String cmdb_ci, String debugFlag)**

Creates an instance of the CIRelatedList class. With no arguments, simply constructs an empty instance.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>The table containing the related list. If many-to-many, the many-to-many table.</td>
</tr>
<tr>
<td>refField</td>
<td>String</td>
<td>The reference field in the table for this related list.</td>
</tr>
<tr>
<td>cmdb_ci</td>
<td>String</td>
<td>The sys_id of the CI this list is related to.</td>
</tr>
<tr>
<td>debugFlag</td>
<td>String</td>
<td>If true, debug logging is enabled.</td>
</tr>
</tbody>
</table>

**CIRelatedList - fromXML(String element)**

Initializes this instance from the specified XML element.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element</td>
<td>String</td>
<td>An XML element</td>
</tr>
</tbody>
</table>

**CIRelatedList - populate()**

Populates the `this.records` field of this instance.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### CIRelatedList - toString(Array result)

Makes a string representation of this instance, pushing each line onto the end of the given result array.

This method is called from CIData.toString(), and assumes related list lines are to be indented two spaces.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>Array</td>
<td>The result to convert</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### CIRelatedList - toXML(String document, String element)

Serializes this instance to XML in the given document and <rl> element.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document</td>
<td>String</td>
<td>The document</td>
</tr>
<tr>
<td>element</td>
<td>String</td>
<td>The rl element</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### CIRelatedList - remove()

Removes all the related list items.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CIRelatedList - xmlToRecord(String records, String element)

Converts a related list from XML to record format.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>records</td>
<td>String</td>
<td>The record to create</td>
</tr>
<tr>
<td>element</td>
<td>String</td>
<td>The XML element</td>
</tr>
</tbody>
</table>

ClassifierProbes

Provides a classifier probe launch facility. Encapsulates an array of probe information records, initialized either from a JavaScript array or the XML serialized version.

Use in any server-side script where you need to define a classifier probe launch facility.

ClassifierProbes - launch()

Launches the next highest priority classifier from those in the list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if a classifier is launched; otherwise, false.</td>
</tr>
</tbody>
</table>

ClassifierProbes - launchSupplementary()

Launches a supplementary probe.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ClassifierProbes - size()**

Returns the probe length.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The probe length</td>
</tr>
</tbody>
</table>

**Client**

Provides methods to add data to the MetricBase database, to execute transforms on the MetricBase database, and to receive the results of the transforms.

The Client class can be used in scoped and global server scripts. When using the Client class, use the sn_clotho namespace identifier.

This class is part of the MetricBase application.

**Scoped Client - Client()**

Creates an instance of the client class to access the MetricBase database.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
var client = new sn_clotho.Client();
```

**Scoped Client - put(DataBuilder metricData)**

Save metric data to the MetricBase database.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metricData</td>
<td>DataBuilder</td>
<td>A DataBuilder object containing metric data.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### CMDBGroupAPI

The CMDBGroupAPI provided methods for performing actions on CMDB groups.

The CMDBGroupAPI is a scoped static class. To use the class you must include the namespace identifier sn_cmdbgroup before the CMDBGroupAPI object. For example:

```javascript
var response = sn_cmdbgroup.CMDBGroupAPI.getManualCIList(groupSysId, false);
```

### CMDBGroupAPI - getAllCI( String groupId, Boolean requireCompleteSet)

Returs all CIs for this group. This includes all manual CIs and the list of CIs from the Query Builder’s saved query.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>String</td>
<td>The sysId of the CMDB group.</td>
</tr>
<tr>
<td>requireCompleteSet</td>
<td>Boolean</td>
<td>When true, returns an empty string if any CIs are filtered out by ACL restrictions.</td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string in the format</td>
</tr>
</tbody>
</table>

```json
{
  'result':false,
  'errors':[ {
    'message':'Group does not exist',
    'error':'GROUP_SYS_ID_IS_NOT_FOUND'
  },
  { } // another error if it exists
},
'partialCIListDueToACLFlag':false,
'idList':['sys_id_1', 'sys_id2']
}
```

Where

- **result** - a boolean flag. When true the method was successful.
- **errors** - a list of errors with a message and error code.
- **partialCIListDueToACLFlag** - a Boolean flag. When true, the idList is incomplete due to an ACL restriction. When false, the idList is complete.
- **idList** - an array of cmdb_ci sys_ids

When not successful, returns one of the errors GROUP_SYS_ID_IS_NOT_FOUND, GROUP_SYS_ID_IS_EMPTY, FAIL_TO_INSERT_GROUP_CI_PAIR, FAIL_TO_INSERT_GROUP_QUERY_ID_PAIR, CI_CAN_NOT_FOUND, SAVED_QUERY_ID_NOT_FOUND, ERROR_DURING_QUERY_BUILDER_PROCESS_QUERY, TIMEOUT_DURING_QUERY_BUILDER_PROCESS_QUERY, NOT_COMPLETE_DURING_QUERY_BUILDER_PROCESS_QUERY, MAX_LIMIT_DURING_QUERY_BUILDER_PROCESS_QUERY, GROUP_API_TIMEOUT, EXCEPTION_FROM_EXECUTE_QUERY, SOME_CI_NOT_VISIBLE_DUE_TO_SECURITY_CONSTRAINT

// Script example:
```javascript
var getAllCIFunc = function(groupSysId) {
  var parser = new JSONParser();
  var response = sn_cmdbgroup.CMDBGroupAPI.getAllCI(groupSysId, false);
  var parsed = parser.parse(response);
  if (parsed.result) {
    gs.print("succeed to retrieve ci list: " + parsed.idList);
  } else {
    gs.print("fail to retrieve list, errors: " + JSON.stringify(parsed.errors));
  }
}
```
var groupExists = "d0d2d25113152200eef2dd828144b0e4";
var groupContainsInvalidSavedQuery = "e685a2c3d7012200de92a5f75e610387";
getAllCIFunc(groupExists);
getAllCIFunc(groupContainsInvalidSavedQuery);

Output: (Line breaks added for formatting.)

succeed to retrieve ci list: 539747cac0a801640163e60735fbbf6e,
3a5dd3dbc0a8ce0100655f1ec66ed42c,6b43105c37301000deeabfc8bcbe5db2,
3a290cc60a0a0bb400000db386af1cf,53958ff0c0a801640171ec76aa0c8f86,
2dfd7c8437201000deeabfc8bcbe5d56
fail to retrieve list, errors: [{"message": "Query Builder process status is failed for processing query id 6d85a683d7012200de92a5f75e6103c0", "error": "ERROR_DURING_QUERY_BUILDER_PROCESS_QUERY"}]

CMDBGroupAPI - getAllCIFromQueryBuilder( String groupld, Boolean requireCompleteSet)

Returns all CIs returned from all saved query builder's query IDs for the specified group.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupld</td>
<td>String</td>
<td>The sysId of the CMDB group.</td>
</tr>
<tr>
<td>requireCompleteSet</td>
<td>Boolean</td>
<td>When true, returns an empty string if any CIs are filtered out by ACL restrictions.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string in the format</td>
</tr>
<tr>
<td></td>
<td>```{ 'result':false,</td>
</tr>
<tr>
<td></td>
<td>'errors':[ {'message':'Group does not exist',</td>
</tr>
<tr>
<td></td>
<td>'error':'GROUP_SYS_ID_IS_NOT_FOUND'},</td>
</tr>
<tr>
<td></td>
<td>{} // another error if it exists</td>
</tr>
<tr>
<td></td>
<td>'partialCIListDueToACLFlag':false,</td>
</tr>
<tr>
<td></td>
<td>'idList':[ 'sys_id_1', 'sys_id2'] `}</td>
</tr>
<tr>
<td></td>
<td>Where</td>
</tr>
<tr>
<td></td>
<td>• result - a boolean flag. When true the method was successful.</td>
</tr>
<tr>
<td></td>
<td>• errors - a list of errors with a message and error code.</td>
</tr>
<tr>
<td></td>
<td>• partialCIListDueToACLFlag - a Boolean flag. When true, the idList is</td>
</tr>
<tr>
<td></td>
<td>incomplete due to an ACL restriction. When false, the idList is</td>
</tr>
<tr>
<td></td>
<td>complete.</td>
</tr>
<tr>
<td></td>
<td>• idList - an array of cmdb_ci sys_ids</td>
</tr>
<tr>
<td></td>
<td>When not successful, returns one of the errors:</td>
</tr>
<tr>
<td></td>
<td>GROUP_SYS_ID_IS_NOT_FOUND,</td>
</tr>
<tr>
<td></td>
<td>GROUP_SYS_ID_IS_EMPTY,</td>
</tr>
<tr>
<td></td>
<td>FAIL_TO_INSERT_GROUP_CI_PAIR,</td>
</tr>
<tr>
<td></td>
<td>FAIL_TO_INSERT_GROUP_QUERY_ID_PAIR,</td>
</tr>
<tr>
<td></td>
<td>CI_CAN_NOT_FOUND,</td>
</tr>
<tr>
<td></td>
<td>SAVED_QUERY_ID_NOT_FOUND,</td>
</tr>
<tr>
<td></td>
<td>ERROR_DURING_QUERY_BUILDER_PROCESS_QUERY,</td>
</tr>
<tr>
<td></td>
<td>TIMEOUT_DURING_QUERY_BUILDER_PROCESS_QUERY,</td>
</tr>
<tr>
<td></td>
<td>NOT_COMPLETE_DURING_QUERY_BUILDER_PROCESS_QUERY,</td>
</tr>
<tr>
<td></td>
<td>MAX_LIMIT_DURING_QUERY_BUILDER_PROCESS_QUERY,</td>
</tr>
<tr>
<td></td>
<td>GROUP_API_TIMEOUT,</td>
</tr>
<tr>
<td></td>
<td>EXCEPTION_FROM_EXECUTE_QUERY,</td>
</tr>
<tr>
<td></td>
<td>SOME_CI_NOT_VISIBLE_DUE_TO_SECURITY_CONSTRAINT</td>
</tr>
</tbody>
</table>

// Script example:
var getAllCIFromQueryBuilderFunc = function(groupSysId) {
  var parser = new JSONParser();
  var response =
    sn_cmdbgroup.CMDBGroupAPI.getAllCIFromQueryBuilder(groupSysId, false);
  var parsed = parser.parse(response);
  if (parsed.result) {
    gs.print("succeed to retrieve ci list: " + parsed.idList);
  } else {
    gs.print("fail to retrieve list, errors: " + JSON.stringify(parsed.errors));
  }
var groupExists = "d0d2d25113152200ef2dd828144b0e4";
var groupContainsInvalidSavedQuery =
  "e685a2c3d7012200de92a5f75e610387";
getAllCIFromQueryBuilderFunc(groupExists);
getAllCIFromQueryBuilderFunc(groupContainsInvalidSavedQuery);

Output: (Line breaks added for formatting.)
succeed to retrieve ci list: 539747cac0a801640163e60735fbbf6e,
3a5dd3d0c0a8ce0100655f1ec66ed42c,3a290cc60a0a0bb400000b8a386a1cf,
53958ff00c0a801640171ec76aa0c8f86
fail to retrieve list, errors: [{"message":"Query Builder
  process status is failed for processing
  query id 6d85a683d7012200de92a5f75e6103c0",
  "error":"ERROR_DURING_QUERY_BUILDER_PROCESS_QUERY"}]

CMDBGroupAPI - getManualCIMList(String groupId, Boolean requireCompleteSet)

Returns the CMDB group's manual CI list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>String</td>
<td>The sysId of the CMDB group.</td>
</tr>
<tr>
<td>requireCompleteSet</td>
<td>Boolean</td>
<td>When true, returns an error string if any CIs are filtered out by ACL restrictions.</td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string in the format</td>
</tr>
<tr>
<td></td>
<td>`{ 'result':false, 'errors':[ {'message':'Group does not exist', 'error':'GROUP_SYS_ID_IS_NOT_FOUND'}], 'partialCILogDueToACLFlag':false, 'idList':['sys_id_1', 'sys_id2'] }</td>
</tr>
</tbody>
</table>

Where

- **result** - a boolean flag. When true the method was successful.
- **errors** - a list of errors with a message and error code.
- **partialCILogDueToACLFlag** - a Boolean flag. When true, the idList is incomplete due to an ACL restriction. When false, the idList is complete.
- **idList** - an array of cmdb_ci sys_ids

When not successful, returns one of the errors `GROUP_SYS_ID_IS_NOT_FOUND`, `GROUP_SYS_ID_IS_EMPTY`, `FAIL_TO_INSERT_GROUP_CI_PAIR`, `FAIL_TO_INSERT_GROUP_QUERY_ID_PAIR`, `CI_CAN_NOT_FOUND`, `SAVED_QUERY_ID_NOT_FOUND`, `ERROR_DURING_QUERY_BUILDER_PROCESS_QUERY`, `TIMEOUT_DURING_QUERY_BUILDER_PROCESS_QUERY`, `NOT_COMPLETE_DURING_QUERY_BUILDER_PROCESS_QUERY`, `MAX_LIMIT_DURING_QUERY_BUILDER_PROCESS_QUERY`, `GROUP_API_TIMEOUT`, `EXCEPTION_FROM_EXECUTE_QUERY`, `SOME_CI_NOT_VISIBLE_DUE_TO_SECURITY_CONSTRAINT`.

```javascript
// Script example for requireCompleteSet being false:
var getManualCILog = function(groupSysId) {
  var parser = new JSONParser();
  var response = sn_cmdbgroup.CMDBGroupAPI.getManualCILog(groupSysId, false);
  var parsed = parser.parse(response);
  if (parsed.result) {
    gs.print("succeed to retrieve ci list: "+ parsed.idList);
  } else {
    gs.print("fail to retrieve list, errors: "+ JSON.stringify(parsed.errors));
  }
```

© 2020 ServiceNow, Inc. All rights reserved.

ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
// create a group in cmdb_group, and add CIs to this group in Edit Manual CI form
var groupExists = "d0d2d25113152200eef2dd828144b0e4";
// use a non-exist group
var groupDoesNotExist = "d0d2d25113152200eef2dd828144b0e4111";
getManualCIList(groupExists);
getManualCIList(groupDoesNotExist);

Output: (Line breaks added for formatting.)

succeed to retrieve ci list:
   6b43105c37301000deeabfc8bcbe5db2,2dfd7c8437201000deeabfc8bcbe5d56
fail to retrieve list, errors:
["message":"Group does not exist","error":"GROUP_SYS_ID_IS_NOT_FOUND"]

// Script example for requireCompleteSet being true
var getManualCIList = function(groupSysId) {
  var response =
  sn_cmdbgroup.CMDBGroupAPI.getManualCIList(groupSysId, true);
  var parsed = parser.parse(response);
  if (parsed.result) {
    gs.print("succeed to retrieve ci list: " + parsed.idList);
  } else {
    gs.print("fail to retrieve list, errors: " + JSON.stringify(parsed.errors));
  }
}

// create a group in cmdb_group, and add CIs to this group in Edit Manual CI form
var groupExists = "d0d2d25113152200eef2dd828144b0e4";
getManualCIList(groupExists);

Output: (Line breaks added for formatting.)

fail to retrieve list, errors:
["message":"Some CI(s) not visible due to security constraint","error":"SOME_CI_NOT_VISIBLE_DUE_TO_SECURITY_CONSTRAINT"]

**CMDBGroupAPI - getSavedQueryIdList(String groupld, Boolean requireCompleteSet)**

Returns the query builder's query IDs for the specified CMDB group.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupld</td>
<td>String</td>
<td>The sysid of the CMDB group.</td>
</tr>
<tr>
<td>requireCompleteSet</td>
<td>Boolean</td>
<td>When true, returns an empty string if any CIs are filtered out by ACL restrictions.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>A JSON formatted string in the format</td>
<td></td>
</tr>
</tbody>
</table>
|            | {
|            |   'result':false,
|            |   'errors':[ {'message':'Group does not exist'},
|            |     'error':'GROUP_SYS_ID_IS_NOT_FOUND'},
|            |   {} // another error if it exists
|            | },
|            | 'partialCIListDueToACLFlag':false,
|            | 'idList':['sys_id_1', 'sys_id2'] }                                                                                                                                                                        |

Where

- result - a boolean flag. When true the method was successful.
- errors - a list of errors with a message and error code.
- partialCIListDueToACLFlag - a Boolean flag. When true, the idList is incomplete due to an ACL restriction. When false, the idList is complete.
- idList - an array of cmdb_ci sys_ids

When not successful, returns one of the errors GROUP_SYS_ID_IS_NOT_FOUND, GROUP_SYS_ID_IS_EMPTY, FAIL_TO_INSERT_GROUP_CI_PAIR, FAIL_TO_INSERT_GROUP_QUERY_ID_PAIR, CI_CAN_NOT_FOUND, SAVED_QUERY_ID_NOT_FOUND, ERROR_DURING_QUERY_BUILDER_PROCESS_QUERY, TIMEOUT_DURING_QUERY_BUILDER_PROCESS_QUERY, NOT_COMPLETE_DURING_QUERY_BUILDER_PROCESS_QUERY, MAX_LIMIT_DURING_QUERY_BUILDER_PROCESS_QUERY, GROUP_API_TIMEOUT, EXCEPTION_FROM_EXECUTE_QUERY, SOME_CI_NOT_VISIBLE_DUE_TO_SECURITY_CONSTRAINT

// Script example:
```javascript
var getSavedQueryIdList = function(groupSysId) {
  var parser = new JSONParser();
  var response = sn_cmdbgroup.CMDBGroupAPI.getSavedQueryIdList(groupSysId,
false);
  var parsed = parser.parse(response);
  if (parsed.result) {
    gs.print("succeed to retrieve saved query id list: " +
parsed.idList);
  } else {
    gs.print("fail to retrieve list, errors: " +
JSON.stringify(parsed.errors));
  }
};
```
var groupExists = "d0d2d25113152200eef2dd828144b0e4";
var groupDoesNotExists = "d0d2d25113152200eef2dd828144b0e4111";
getSavedQueryIdList(groupExists);
getSavedQueryIdList(groupDoesNotExists);

Output: (Line breaks added for formatting.)

succeed to retrieve saved query id list:
  5d498532d7c12200dc92a5f75e6103ce
fail to retrieve list, errors:
  [{"message":"Group does not exist","error":"GROUP_SYS_ID_IS_NOT_FOUND"}]

CMDBGroupAPI - setManualCIList(String groupId, String ciSysIds)

Sets the manual CI list for the specified group. The existing manual CI list is overwritten. CI sysIds not found in the cmdb_ci table are ignored.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>String</td>
<td>The sysId of the CMDB group.</td>
</tr>
<tr>
<td>ciSysIds</td>
<td>String</td>
<td>Comma separated list of CI sysIds.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td>JSON Example</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>String</td>
<td>A JSON formatted string in the format</td>
<td>{ 'result':false, 'errors':[ { 'message':'Group does not exist', 'error':'GROUP_SYS_ID_IS_NOT_FOUND' }, {}, // another error if it exists }, 'partialCIListDueToACLFlag':false, 'idList':['sys_id_1', 'sys_id2'] }</td>
</tr>
</tbody>
</table>

Where
- result - a boolean flag. When true the method was successful.
- errors - a list of errors with a message and error code.
- partialCIListDueToACLFlag - a Boolean flag. When true, the idList is incomplete due to an ACL restriction. When false, the idList is complete.
- idList - an array of cmdb_ci sys_ids

When not successful, returns one of the errors GROUP_SYS_ID_IS_NOT_FOUND, GROUP_SYS_ID_IS_EMPTY, FAIL_TO_INSERT_GROUP_CI_PAIR, FAIL_TO_INSERT_GROUP_QUERY_ID_PAIR, CI_CAN_NOT_FOUND, SAVED_QUERY_ID_NOT_FOUND, ERROR_DURING_QUERY_BUILDER_PROCESS_QUERY, TIMEOUT_DURING_QUERY_BUILDER_PROCESS_QUERY, NOT_COMPLETE_DURING_QUERY_BUILDER_PROCESS_QUERY, MAX_LIMIT_DURING_QUERY_BUILDER_PROCESS_QUERY, GROUP_API_TIMEOUT, EXCEPTION_FROM_EXECUTE_QUERY, SOME_CI_NOT_VISIBLE_DUE_TO_SECURITY_CONSTRAINT

// Script example:
var setManualCIListFunc = function(groupSysId, manualCIList) {
    var parser = new JSONParser();
    var response = sn_cmdbgroup.CMDBGroupAPI.setManualCIList(groupSysId, manualCIList);
    var parsed = parser.parse(response);
    if (parsed.result) {
        gs.print("succeed to set manual ci list");
    } else {
        gs.print("fail to set manual ci list, errors: " + JSON.stringify(parsed.errors));
    }
}
var group = "d0d2d25113152200eef2dd828144b0e4";
var groupDoesNotExist = "1234";
var manualCIList = "b4fd7c8437201000deeabfc8bcbe5dc1,
affd3c8437201000deeabfc8bcbe5dc3";
setManualCIListFunc(group, manualCIList);
setManualCIListFunc(groupDoesNotExist, manualCIList);

Output: (Line breaks added for formatting.)
succeed to set manual ci list
fail to set manual ci list, errors: [{"message":"Group does not exist","error":
"GROUP_SYS_ID_IS_NOT_FOUND"}]

CMDBGroupAPI - setSavedQueryIdList(String groupId, String queryIds)
Sets the saved query ID list for the specified group. The existing query ID list is overwritten. Query sysIds not found in the qb_saved_query table are ignored.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>String</td>
<td>The sysId of the CMDB group.</td>
</tr>
<tr>
<td>queryIds</td>
<td>String</td>
<td>Comma separated list of saved query sysIds.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string in the format</td>
</tr>
<tr>
<td></td>
<td>{ 'result':false, 'errors':[ { 'message': 'Group does not exist', 'error': 'GROUP.Sys_ID.IS.NOT.FOUND'}, { } // another error if it exists }, 'partialCIListDueToACLFlag':false, 'idList':['sys_id_1', 'sys_id2'] }</td>
</tr>
</tbody>
</table>

Where

- result - a boolean flag. When true the method was successful.
- errors - a list of errors with a message and error code.
- partialCIListDueToACLFlag - a Boolean flag. When true, the idList is incomplete due to an ACL restriction. When false, the idList is complete.
- idList - an array of cmdb_ci sys_ids

When not successful, returns one of the errors GROUP_SYS_ID_IS_NOT_FOUND, GROUP_SYS_ID_IS_EMPTY, FAIL_TO_INSERT_GROUP_CI_PAIR, FAIL_TO_INSERT_GROUP_QUERY_ID_PAIR, CI_CAN_NOT_FOUND, SAVED_QUERY_ID_NOT_FOUND, ERROR_DURING_QUERY_BUILDER_PROCESS_QUERY, TIMEOUT_DURING_QUERY_BUILDER_PROCESS_QUERY, NOT_COMPLETE_DURING_QUERY_BUILDER_PROCESS_QUERY, MAX_LIMIT_DURING_QUERY_BUILDER_PROCESS_QUERY, GROUP_API_TIMEOUT, EXCEPTION_FROM_EXECUTE_QUERY, SOME_CI_NOT_VISIBLE_DUE_TO_SECURITY_CONSTRAINT

// Script example:
var setSavedQueryIdListFunc = function(groupSysId, queryIdList) {
    var parser = new JSONParser();
    var response =
        sn_cmdbgroup.CMDBGroupAPI.setSavedQueryIdList(groupSysId, queryIdList);
    var parsed = parser.parse(response);
    if (parsed.result) {
        gs.print("succeed to set saved query id list");
    } else {
        gs.print("fail to set saved query id list, errors: " +
            JSON.stringify(parsed.errors));
    
© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
CMDBTransformUtil

The CMDBTransformUtil class uses the Identification and Reconciliation framework to minimize creation of duplicate CIs and to reconcile CI attributes by only accepting information from authorized sources when updating the CMDB in onBefore transform map scripts. This class cannot be used in other scripts.

CMDBTransformUtil - CMDBTransformUtil()

Creates a CMDBTransformUtil object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

var cmdbUtil = new CMDBTransformUtil();

CMDBTransformUtil - getError()

Use this method in onBefore transform map scripts to get the error when the identifyAndReconcile() method was called.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>The error message from identifyAndReconcile().</td>
<td></td>
</tr>
</tbody>
</table>

```java
// add this code to the onBefore transform map script
// Call CMDB API to do Identification and Reconciliation of current row
var cmdbUtil = new CMDBTransformUtil();
cmdbUtil.setDataSource('ImportSet');
cmdbUtil.identifyAndReconcile(source, map, log);
ignore = true;

if (cmdbUtil.hasError()) {
    var errorMessage = cmdbUtil.getError();
    log.error(errorMessage);
} else {
    log.info('IE Output Payload: ' + cmdbUtil.getOutputPayload());
    log.info('Imported CI: ' + cmdbUtil.getOutputRecordSysId());
}
```

**Scoped equivalent**

There is no scoped equivalent for this method.

**CMDBTransformUtil - getOutputPayload()**

Use this method in onBefore transform map scripts to get the JSON payload after the identifyAndReconcile() method is called.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| String  | A JSON formatted string that is a list of results for the configuration items in the input string. Each result string is in the format `{items: [{}], relations: [{}]},` where each item within the items and relations lists contains name-value pairs. The possible name-value pairs within the items list are:  
  - className - the sys_class_name for the CI that was updated or created.  
  - operation, which is one of INSERT, UPDATE, UPDATE_WITH_UPGRADE, UPDATE_WITH_DOWNGRADE, UPDATE_WITH_SWITCH, DELETE, NO_CHANGE  
  - sysId - the sys_id of the CI that was updated or created.  
  - relatedSysIds - a list of sys_id values of CIs used during lookup based identification.  
  - maskedAttributes - a list of attributes whose update by a non-authoritative data source gets skipped as defined by the Reconciliation Rules.  
  - identifierEntrySysId - sys_id of identifier entry used during matching.  
  - errors - a list of errors in the format of (error, message string), where error can be ABANDONED, INVALID_INPUT_DATA, IDENTIFICATION_RULE_MISSING, IDENTIFICATION_RULE_FOR_LOOKUP_MISSING, NO_LOOKUP_RULES_FOR_DEPENDENT_CI, NO_CLASS_NAME_FOR_INDEPENDENT_CI, MISSING_DEPENDENCY, MULTIPLE_DEPENDENCIES, MULTIPLE_DUPPLICATE_RECORDS, RELATION_CHAIN_ENDS_AT_QUALIFIER, QUALIFICATION_LOOP, TYPE_CONFLICT_IN_QUALIFICATION, MULTI_MATCH, REQUIRED_ATTRIBUTE_EMPTY, RECLASSIFICATION_NOT_ALLOWED  
  - duplicateIndices - a list of indexes of items that are duplicates of the current item.  
  - identificationAttempts - a list of attempts in the format of (attributes, identifierName, attemptResult, searchOnTable) where  
    - attributes - the attributes of identifier entry used during identification  
    - identifierName - the CI identifier to which this identifier entry belongs  
    - attemptResult - one of SKIPPED, NO_MATCH, MATCHED, MULTI_MATCH  
    - searchOnTable - the table searched during the identification process. |
// Call CMDB API to do Identification and Reconciliation of current row
var cmdbUtil = new CMDBTransformUtil();
cmdbUtil.setDataSource('ImportSet');
cmdbUtil.identifyAndReconcile(source, map, log);
ignore = true;

if (cmdbUtil.hasError()) {
    var errorMessage = cmdbUtil.getError();
    log.error(errorMessage);
} else {
    log.info('IE Output Payload: ' + cmdbUtil.getOutputPayload());
    log.info('Imported CI: ' + cmdbUtil.getOutputRecordSysId());
}

Scoped equivalent

There is no scoped equivalent for this method.

CMDBTransformUtil - getOutputRecordSysId()

Use this method in onBefore transform map scripts to get the sys_id of the CI that was inserted/updated.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the CI that was inserted/updated.</td>
</tr>
</tbody>
</table>

// add this code to the onBefore transform map script
// Call CMDB API to do Identification and Reconciliation of current row
var cmdbUtil = new CMDBTransformUtil();
cmdbUtil.setDataSource('ImportSet');
cmdbUtil.identifyAndReconcile(source, map, log);
ignore = true;

if (cmdbUtil.hasError()) {
    var errorMessage = cmdbUtil.getError();
    log.error(errorMessage);
} else {
    log.info('IE Output Payload: ' + cmdbUtil.getOutputPayload());
    log.info('Imported CI: ' + cmdbUtil.getOutputRecordSysId());
}
Scoped equivalent

There is no scoped equivalent for this method.

CMDBTransformUtil - hasError()

Use this method in onBefore transform map scripts to determine if there was an error when the identifyAndReconcile() method was called.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the identifyAndReconcile() method had an error.</td>
</tr>
</tbody>
</table>

```
// add this code to the onBefore transform map script
// Call CMDB API to do Identification and Reconciliation of current row
var cmdbUtil = new CMDBTransformUtil();
cmdbUtil.setDataSource('ImportSet');
cmdbUtil.identifyAndReconcile(source, map, log);
ignore = true;
if (cmdbUtil.hasError()) {
    var errorMessage = cmdbUtil.getError();
    log.error(errorMessage);
} else {
    log.info('IE Output Payload: ' +
             cmdbUtil.getOutputPayload());
    log.info('Imported CI: ' +
             cmdbUtil.getOutputRecordSysId());
}
```

Scoped equivalent

There is no scoped equivalent for this method.

CMDBTransformUtil - identifyAndReconcile (Object source, Object map, Object log)

Use this method in onBefore transform map scripts to insert or update a configuration item to the CDMB. Using this method instead of relying on the transform to insert or update the configuration item record reduces duplicate entries in the CDMB.
This method is used in the onBefore script. Set `ignore = true` to prevent the transform from adding or updating the record a second time.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Object</td>
<td>This is the global object available in transform map scripts that holds the incoming record.</td>
</tr>
<tr>
<td>map</td>
<td>Object</td>
<td>This is the map object available in transform map scripts that holds the current transform map record.</td>
</tr>
<tr>
<td>log</td>
<td>Object</td>
<td>This is the log object available in transform map scripts.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
// add this code to the onBefore transform map script
// Call CMDB API to do Identification and Reconciliation of current row
var cmdbUtil = new CMDBTransformUtil();
cmdbUtil.identifyAndReconcile(source, map, log);
ignore = true;
```

**CMDBTransformUtil - logTransformStats( Object log)**

Use this method in onComplete transform map scripts to log the number of CIs inserted, updated, skipped, or had errors.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log</td>
<td>Object</td>
<td>This is the log object available in transform map scripts.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
Scoped equivalent

There is no scoped equivalent for this method.

**CMDBTransformUtil - setDataSource(String source)**

Use this method in onBefore transform map scripts to set the data source to be used when the `identifyAndReconcile()` method is called. If this method is not called, the default value 'ImportSet' is used.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>String</td>
<td>The data source for the source CI record.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
// add this code to the onBefore transform map script
// Call CMDB API to do Identification and Reconciliation of current row
var cmdbUtil = new CMDBTransformUtil();
cmdbUtil.setDataSource('ImportSet');
cmdbUtil.identifyAndReconcile(source, map, log);
ignore = true;
if (cmdUtil.hasError()) {
    var errorMessage = cmdUtil.getError();
    log.error(errorMessage);
} else {
    log.info('IE Output Payload: ' +
             cmdUtil.getOutputPayload());
    log.info('Imported CI: ' +
             cmdUtil.getOutputRecordSysId());
}
```

Scoped equivalent

There is no scoped equivalent for this method.

**CMDBUtil**

The CMDBUtil API provides utility methods for creating and managing table relationships in the configuration management database (CMDB) and managing CMDB baselines.

CMDBUtil is a JavaScript-accessible ScriptableObject. The CMDBUtil API has dynamic and static methods. You access dynamic methods by creating a SNC.CMDBUtil object. You access static methods by using SNC.CMDBUtil global object to call the methods.
This example creates a CMDBUtil object.

```javascript
var cu = new SNC.CMDBUtil();
cu.baselineProposedChangesGenDIFF(current, action.get('sysparm_changeset'));
```

This example calls a static method.

```javascript
var output = SNC.CMDBUtil.getAllChildrenOfAsCommaList('cmdb_ci_computer');
```

Use these methods to manage CMDB table relationships and baselines.

**CMDBUtil - baselineProposedChangesApplyChanges(GlideRecord proposedChanges)**

Applies proposed changes in a task_ci record that represent an update set for all CIs associated with the task.

This is a dynamic method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposedChanges</td>
<td>GlideRecord</td>
<td>The collection of proposed changes in the change request (CHG).</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
//Where current is a GlideRecord and action is the current UI action
var base = new SNC.CMDBUtil();
base.baselineProposedChangesApplyChanges(current);
```

**CMDBUtil - baselineProposedChangesGenDiff(GlideRecord current, String changeSet)**

Generates the XML for proposed changes diff, and adds it to the corresponding task_ci record.

This is a dynamic method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current</td>
<td>GlideRecord</td>
<td>The GlideRecord that contains the collection of proposed changes.</td>
</tr>
<tr>
<td>changeSet</td>
<td>String</td>
<td>The sysid of the task_ci record that represents the change set to use.</td>
</tr>
</tbody>
</table>
// Where current is a GlideRecord and action is the current UI
action
var base = new SNC.CMDBUtil();
base.baselineProposedChangesGenDIFF(current,
action.get('sysparm_changeset'));

**CMDBUtil - bootstrap(String dictFile)**

Bootstraps the environment by creating the database, creating the system dictionary table, and having the system dictionary table describe itself.

This is a static method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dictFile</td>
<td>String</td>
<td>The file name including path, of the dictionary to load.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CMDBUtil - createCIRelationship(String tableName, String parentField, String childField, String parentDesc, String childDesc)**

Creates the specified CI relationship using the specified invocation parameters.

If called without the first parameter, passing only four parameters, defaults to the CI Relationship (cmdb_rel_ci) table.

This is a dynamic method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table name.</td>
</tr>
<tr>
<td>parentField</td>
<td>String</td>
<td>The parent field.</td>
</tr>
<tr>
<td>childField</td>
<td>String</td>
<td>The child field.</td>
</tr>
<tr>
<td>parentDesc</td>
<td>String</td>
<td>The parent relationship descriptor.</td>
</tr>
<tr>
<td>childDesc</td>
<td>String</td>
<td>The child relationship descriptor.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CMDBUtil - getAllChildrenOfAsCommaList(String baseTable)**

Gets all the child tables of the specified table as a comma-separated list.
This is a static method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseTable</td>
<td>String</td>
<td>The base table name.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A comma-separated list of sys_ids of tables extending the base table.</td>
</tr>
</tbody>
</table>

```java
//Where cmdv_ci_computer is a table.
var output = SNC.CMDBUtil.getAllChildrenOfAsCommaList('cmdv_ci_computer');
gs.print(output);
```

**CMDBUtil - getCMDBViews()**

Gets all the records in the CMDB View (cmdv_view) table.
This is a dynamic method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>The records in the CMDB view table.</td>
</tr>
</tbody>
</table>

**CMDBUtil - getTables0(String tableName)**

Gets a list of all the parents of a table.
This is a static method.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table name</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>List of tables that are parents of the specified table.</td>
</tr>
</tbody>
</table>

CMDBUtil - isExcludedFromBR(String className)

Determines whether a CI class is defined in the Business Rule Exclusion Lists (cmdb_business_rule_exclusions) table. Use this method in an advanced condition to prevent a business rule from executing on excluded CI classes.

For example, the Create Asset on insert business rule uses this method in the advanced condition to prevent the rule from creating assets for CI classes such as cmdb_ci_qualifier, cmdb_ci_endpoint, cmdb_ci_storage_volume, and cmdb_ci_vcenter_datastore_disk. Define classes for exclusion in the Business Rule Exclusion Lists (cmdb_business_rule_exclusions) table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>className</td>
<td>String</td>
<td>Name of the class to check for exclusion.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the class is defined in the Business Rule Exclusion Lists (cmdb_business_rule_exclusions) table; otherwise, false.</td>
</tr>
</tbody>
</table>

```
//Returns true if the cmdb_ci_endpoint class is defined
//in the Business Rule Exclusion Lists table
var exclusion = SNC.CMDBUtil.isExcludedFromBR("cmdb_ci_endpoint");

//Advanced condition that prevents business rules from executing on
//classes in the Business Rule Exclusion Lists table
(!SNC.CMDBUtil.isExcludedFromBR(current.getTableName()))
```

CMDBUtil - processCIChange(GlideRecord event, GlideRecord target)

Wraps the call to RelationshipEventProcessor(), which processes any changes to CI relationships.
This is a dynamic method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>GlideRecord</td>
<td>The event record.</td>
</tr>
<tr>
<td>target</td>
<td>GlideRecord</td>
<td>The target record.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CMDBUtil - processRelChange(GlideRecord event, GlideRecord current, String relType, String triplet)**

Wraps the call to `RelationshipEventProcessor()`, which processes any changes to CI relationships, with the specified type and triplet.

This is a dynamic method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>GlideRecord</td>
<td>The event record</td>
</tr>
<tr>
<td>current</td>
<td>GlideRecord</td>
<td>The current record, which is either the relation record or a user record if the current process is a deletion.</td>
</tr>
<tr>
<td>relType</td>
<td>String</td>
<td>The type of relation that changed.</td>
</tr>
<tr>
<td>triplet</td>
<td>String</td>
<td>The child, parent, and class name from the relation that changed.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CMDBUtil - removeCIRelationship(String tableName, String parentField, String childField, String parentDesc, Object childDesc)**

Deletes the specified CI relationship.

If called without the first parameter, passing only four parameters, defaults to the CI Relationship (`cmdb_rel_ci`) table.

This is a dynamic method.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table name</td>
</tr>
<tr>
<td>parentField</td>
<td>String</td>
<td>The parent field</td>
</tr>
<tr>
<td>childField</td>
<td>String</td>
<td>The child field</td>
</tr>
<tr>
<td>parentDesc</td>
<td>String</td>
<td>The parent relationship descriptor</td>
</tr>
<tr>
<td>childDesc</td>
<td>Object</td>
<td>The child relationship descriptor</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**CMDBUtil - reParentTable(String table, String oldParent, String newParent)**

Changes the parent of the given table to the new parent.

This is a static method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>The table to re-parent</td>
</tr>
<tr>
<td>oldParent</td>
<td>String</td>
<td>The old parent</td>
</tr>
<tr>
<td>newParent</td>
<td>String</td>
<td>The new parent</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True, if successful; otherwise, false.</td>
</tr>
</tbody>
</table>

**ConnectActionResponse**

The ConnectActionResponse API provides a method to create a GlideRecord.

ConnectActionResponse methods are accessed through a global object (response) that is available only in Connect action scripts.

**ConnectActionResponse - newRecord(String table, Object values, String view)**

Create a GlideRecord in the specified table with the specified values.

Opens a form in the UI based upon the table, which replaces the Table and Document fields in the Live Group Profile record for the associated conversation.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>Name of the table where the record is to be added.</td>
</tr>
<tr>
<td>values</td>
<td>Object</td>
<td>An object of name value pairs where the names are field names in the table and the values are the field values.</td>
</tr>
<tr>
<td>view</td>
<td>String</td>
<td>The form view to load. This parameter is optional.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
response.newRecord("incident", {
  short_description: conversation.document.short_description || "",
  caller_id: conversation.document.opened_by
});
```

### ConnectionInfo

Use ConnectionInfo API to get connection attribute information through the connection and credential alias.

You can use this API in scoped applications, or within the global scope. In scoped scripts, use the `sn_cc` namespace identifier.

This function retrieves connection attribute information identified by the given connection and credential alias.

```javascript
var provider = new sn_cc.ConnectionInfoProvider();

// get a jdbc connection in the current domain with the alias ID
// "6219afbf9f03320021dd7501942e70fc"
var connectionInfo = provider.getConnectionInfo("6219afbf9f03320021dd7501942e70fc");
if (connectionInfo != null) {
  // get data map
  var datamap = connectionInfo.getDataMap();
  gs.info(datamap["name"]);
  gs.info(datamap["connection_url"]);

  // get the same values using getAttribute
  gs.info(connectionInfo.getAttribute("name"));
  gs.info(connectionInfo.getAttribute("connection_url"));

  // get credential attributes
  gs.info(connectionInfo.getCredentialAttribute("user_name"));
```
```javascript
gs.info(connectionInfo.getCredentialAttribute("password"));

// get extended attributes
var extendedAttributes = connection.getExtendedAttributes();
gs.info(extendedAttributes["name1"]);
}

// get a jdbc connection in the ACME domain with the alias ID
//      "cd5923ff9f03320021dd7501942e70bb"
connectionInfo = provider.getConnectionInfoByDomain("cd5923ff9f03320021dd7501942e70bb",
     "c90d4b084a362312013398f051272c0d");
if (connectionInfo != null) {
    // get data map
    var datamap = connectionInfo.getDataMap();
gs.info(datamap["name"]);
}
```

### Scoped ConnectionInfo - getAttribute(String name)

Returns the value of a connection info attribute with the specified name.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
// get the same values using getAttribute
gs.info(connectionInfo.getAttribute("name"));
gs.info(connectionInfo.getAttribute("connection_url"));
```

### Scoped ConnectionInfo - getCredentialAttribute(String name)

Returns the value of credential attributes for a specified connection.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
// get credential attributes
gs.info(connectionInfo.getCredentialAttribute("user_name"));
gs.info(connectionInfo.getCredentialAttribute("password"));```

**Scoped ConnectionInfo - getDataMap(String)**

Returns the connection attributes as a collection of key-value pairs.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
// get data map
var datamap = connectionInfo.getDataMap();
gs.info(datamap["name"]);
gs.info(datamap["connection_url"]);```

**Scoped ConnectionInfo - getExtendedAttributes(String)**

Returns the extended attributes as a collection of key-value pairs.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
// get extended attributes
var extendedAttributes = connection.getExtendedAttributes();```
ConnectionInfoProvider

Use ConnectionInfoProvider API to select connection information through the connection alias. You can use this API in scoped applications, or within the global scope. In scoped scripts, use the `sn_cc` namespace identifier.

This function retrieves connection information identified by the given connection alias.

```js
var provider = new sn_cc.ConnectionInfoProvider();

// get a jdbc connection in the current domain with the alias ID
//   "6219afbf9f03320021dd7501942e70fc"
var connectionInfo =
    provider.getConnectionInfo("6219afbf9f03320021dd7501942e70fc");
if (connectionInfo != null) {
    // get data map
    var datamap = connectionInfo.getDataMap();
    gs.info(datamap["name"]);
    gs.info(datamap["connection_url"]);

    // get the same values using getAttribute
    gs.info(connectionInfo.getAttribute("name"));
    gs.info(connectionInfo.getAttribute("connection_url"));

    // get credential attributes
    gs.info(connectionInfo.getCredentialAttribute("user_name"));
    gs.info(connectionInfo.getCredentialAttribute("password"));

    // get extended attributes
    var extendedAttributes = connection.getExtendedAttributes();
    gs.info(extendedAttributes["name1"]);
}

// get a jdbc connection in the ACME domain with the alias ID
//   "cd5923ff9f03320021dd7501942e70bb"
connectionInfo =
    provider.getConnectionInfoByDomain("cd5923ff9f03320021dd7501942e70bb",
"c90d4b084a362312013398f051272c0d");
if (connectionInfo != null) {
    // get data map
    var datamap = connectionInfo.getDataMap();
    gs.info(datamap["name"]);
}
```

**ConnectionInfoProvider - ConnectionInfoProvider()**

Uses ConnectionInfoProvider() to select connection information through the connection alias.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Scoped ConnectionInfoProvider - getConnectionInfo(String aliasID)**

This function retrieves a ConnectionInfo object identified by the given aliasID in the current domain.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aliasID</td>
<td>String</td>
<td>The sys_id of a connection alias.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionInfo</td>
<td>Information about the connection.</td>
</tr>
</tbody>
</table>

```javascript
var provider = new sn_cc.ConnectionInfoProvider();

// get a jdbc connection in the current domain with the alias ID
// "6219afbf9f03320021dd7501942e70fc"
var connectionInfo = provider.getConnectionInfo("6219afbf9f03320021dd7501942e70fc");
```

**Scoped ConnectionInfoProvider - getConnectionInfoByDomain(String aliasID, String domainID)**

This function retrieves a ConnectionInfo object identified by the given aliasID for a specific domain.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aliasID</td>
<td>String</td>
<td>The sys_id of a connection alias.</td>
</tr>
<tr>
<td>domainID</td>
<td>String</td>
<td>The sys_id of a domain or global.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionInfo</td>
<td>Connection information.</td>
</tr>
</tbody>
</table>

```javascript
var provider = new sn_cc.ConnectionInfoProvider();

// get a jdbc connection in the ACME domain with the alias ID
// "cd5923ff9f03320021dd7501942e70bb"
connectionInfo = provider.getConnectionInfoByDomain("cd5923ff9f03320021dd7501942e70bb");
```
**Conversation**

Conversation API enables you to create or modify Connect conversations.

To use this class in a scoped application, use the sn_connect namespace identifier. The Connect Scriptable APIs plugin (ID: com.glide.connect.scriptable) should be enabled to access the Conversation API.

**Scoped Conversation - get(String sysID)**

Returns an existing Connect conversation by sys_id.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysID</td>
<td>String</td>
<td>The sys_id of the conversation record.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Conversation object</td>
</tr>
</tbody>
</table>

```javascript
var conversation = sn_connect.Conversation.get("27b9844c1385030034bb58a12244b037");
```

**Scoped Conversation - create(String name)**

Creates a Connect conversation.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Create a conversation with a specific name.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| type | String | Include a specific conversation type. The type is determined by the type choice list. The base system includes the following type options:  
• connect  
• support  
• group  
• peer  
• qanda  
• team |

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Scriptable Conversation</td>
</tr>
</tbody>
</table>

```javascript
var conversation = sn_connect.Conversation.create({
    name: "Hello world",
    type: "connect"
});
```

**Scoped Conversation - addSubscriber(String sysID)**

Adds a user to a conversation.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysID</td>
<td>String</td>
<td>The sys_ID of the user you want to add to a conversation.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var conversation = sn_connect.Conversation.get("7caf4911309030034bb58a12244b06c");
conversation.addSubscriber("a8f98bb0eb32010045e1a5115206fe3a");
```
Scoped Conversation - removeSubscriber(String sysID)

Removes a user from a conversation.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysID</td>
<td>String</td>
<td>The sys_id of the user you want to remove from a conversation.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var conversation = sn_connect.Conversation.get("7caf49111309030034bb58a12244b06c");
conversation.removeSubscriber("a8f98bb0eb32010045e1a5115206fe3a");
```

Scoped Conversation - sendMessage(String body, String field)

Sends a message to a conversation.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body</td>
<td>String</td>
<td>The main text of the message.</td>
</tr>
<tr>
<td>Field</td>
<td>String</td>
<td>The field you want the message to appear as. Only use this option if adding a message to a record conversation. Choose from work_notes, comments, or system. Using system as the field treats the message as a system message.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var conversation = sn_connect.Conversation.get("2064fa3919010300964f5270e9840fbb");
conversation.sendMessage(body: "Hello world", field: "work_notes");
```
CriteriaEvaluator

Implements a general-purpose criteria evaluator.

The general use pattern is to construct the class, add as many criteria as needed, then evaluate.

CriteriaEvaluator - addCriterion(String lhComparand, String operator, String rhComparand)

Adds the criteria to be evaluated.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lhComparand</td>
<td>String</td>
<td>The left hand comparand. Must be a named value in the left hand values input into evaluate().</td>
</tr>
<tr>
<td>operator</td>
<td>String</td>
<td>The operator. Must be one of: equals, starts with, contains, does not contain, ends with, or regex matches.</td>
</tr>
<tr>
<td>rhComparand</td>
<td>String</td>
<td>The right hand comparand. Can be evaluated either as one of the right hand values input to evaluate() or as a literal string.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

CriteriaEvaluator - evaluate(Object lhValues, String rhValues, Boolean any)

Evaluates the criteria.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lhValues</td>
<td>Object</td>
<td>A hashmap of name-value pairs to evaluate the left hand comparands of the criteria.</td>
</tr>
<tr>
<td>rhValues</td>
<td>String</td>
<td>Optional right hand values to use instead of the literal value in evaluating the right hand comparands.</td>
</tr>
</tbody>
</table>
### CustomEvent

You can use CustomEvent API to show qualified embedded help in the right sidebar. See [Use embedded help qualifiers](#) for more information.

#### CustomEvent - fireAll(String event, String qualifier)

Show the embedded-help content specified by the qualifier parameter in the right sidebar. Before using the `fireAll()` method, you must have created the Embedded Help qualifier and help content.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>String</td>
<td>The event to send. Must be the string &quot;embedded_help:load_embedded_help&quot;</td>
</tr>
<tr>
<td>qualifier</td>
<td>String</td>
<td>The qualifier name created in the Embedded Help application.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var qualifier = 'your-EH-qualification';
CustomEvent.fireAll("embedded_help:load_embedded_help", qualifier);
```

### Data

A Data object contains the results of transform performed by a `sn_clotho.Client.transform()` method.
Do not use a constructor to create an instance of this class, instead use the object returned by the
`sn_clotho.Client.transform()` method.

The Data class can be used in scoped and global server scripts. When using the Data class, use
the `sn_clotho` namespace identifier.

This class is part of the MetricBase application.

**Scoped Data - getEnd()**

Returns the end time for data in the Data object.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>GlideDateTime</td>
</tr>
</tbody>
</table>

**Scoped Data - getLabel()**

Returns the label assigned by the `sn_clotho.ClothoTransform.label()` method.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>String</td>
</tr>
</tbody>
</table>

**Scoped Data - getMetricName()**

Returns the name of the metric of the data series. Returns null when the data object is associated
with multiple data series.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.

ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Name of the metric field. Returns null when the data object is associated with multiple data series.</td>
</tr>
</tbody>
</table>

### Scoped Data - getPeriod()

Returns the time period in milliseconds.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The elapsed time in seconds.</td>
</tr>
</tbody>
</table>

### Scoped Data - getStart()

Returns the start time for data in the Data object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDateTime</td>
<td>The time for the first data point.</td>
</tr>
</tbody>
</table>

### Scoped Data - getSubject()

Returns the subject of the data series. Returns null when the data object is associated with multiple data series.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The subject field value of the subject GlideRecord. This is generally the sys_id of the subject GlideRecord.</td>
</tr>
</tbody>
</table>

Scoped Data - getTableName()

Returns the name of the table assigned in the DataSelector class constructor. Returns null when the data object is associated with multiple data series.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Table name. Returns null when the data object is associated with multiple data series.</td>
</tr>
</tbody>
</table>

Scoped Data - getValues()

Returns an array of values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of numbers.</td>
</tr>
</tbody>
</table>

Scoped Data - size()

Returns the number of values in the Data object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of values in the object.</td>
</tr>
</tbody>
</table>

**DataBuilder**

Use the DataBuilder class to create a series of data points for a metric. Use the `sn_clotho.Client.put()` method to save the values.

The DataBuilder class can be used in scoped and global server scripts. When using the DataBuilder class, use the `sn_clotho` namespace identifier.

This class is part of the MetricBase application.

**Scoped DataBuilder - add(GlideDateTime start, Array value)**

Add a series of data points to the DataBuilder object. Each data point is a time stamp and a value.

Uses the start parameter and the retention policy collection period to calculate the time stamp for each value in the array. The first value has the start parameter as the time stamp. This method does not save the data in the MetricBase database. Use the `sn_clotho.Client.put()` method to save the values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>GlideDateTime</td>
<td>The time stamp for the first data point. Subsequent time stamps are calculated using the retention policy collection period.</td>
</tr>
<tr>
<td>value</td>
<td>Array</td>
<td>An array of numbers.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataBuilder</td>
<td>The same DataBuilder object.</td>
</tr>
</tbody>
</table>

```javascript
var points = [7, 0.5, 273];
var dataBuilder = new sn_clotho.DataBuilder(gr, 'cpu_percentage');
// this creates a GlideDateTime object set to the current date and time
var time = new GlideDateTime();
dataBuilder.add(time, points);
```
Scoped DataBuilder - add(GlideDateTime start, Number value)

Adds a data point to the DataBuilder object. Each data point is a time stamp and a value. This method does not save the data point in the metric. Use the sn_clotho.Client.put() method to save the values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>GlideDateTime</td>
<td>The time stamp for the data point.</td>
</tr>
<tr>
<td>value</td>
<td>Number</td>
<td>The value of the data point.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataBuilder</td>
<td>The DataBuilder object.</td>
</tr>
</tbody>
</table>

```javascript
var dataBuilder = new sn_clotho.DataBuilder(gr, 'cpu_percentage');
// this creates a GlideDateTime object set to the current date and time
var time = new GlideDateTime();
dataBuilder.add(time, 0.6);
```

Scoped DataBuilder - DataBuilder(Object glideRecord, String metric)

Creates an instance of the DataBuilder class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>glideRecord</td>
<td>Object</td>
<td>GlideRecord from which to obtain the domain.</td>
</tr>
<tr>
<td>metric</td>
<td>String</td>
<td>The field name of the metric.</td>
</tr>
</tbody>
</table>

```javascript
// Where cpu_percentage is the name of the metric
var dataBuilder = new sn_clotho.DataBuilder(gr, 'cpu_percentage');
```

DateTimeUtils

DateTimeUtils class is a collection of date/time functions.

The DateTimeUtils class is provided via the Script Include DateTimeUtils.

The DateTimeUtils class can be used in any server-side script, and is available through GlideAjax.
DateTimeUtils - int8ToGlideDateTime(Number int64)

Converts Microsoft AD integer8 DateTime format into GlideDateTime format. Integer8 is also known as Microsoft Filetime format. This method is commonly used when importing AD user's date fields, such as Expiration Date.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int64</td>
<td>Number</td>
<td>A 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC).</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDateTime</td>
<td>A GlideDateTime object set to the Integer8 date and time.</td>
</tr>
</tbody>
</table>

```javascript
//convert and set account expiration date from AD
//this is an example that could be used in an LDAP import transform map to import the LDAP account
//expires attribute to a customer created u_account_expires GlideDateTime field
var dtUtil = new DateTimeUtils();
target.u_account_expires = dtUtil.int8ToGlideDateTime(source.u_accountexpires);
```

DateTimeUtils - msToGlideDateTime(Number milliseconds)

Converts milliseconds to a GlideDateTime object

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>Number</td>
<td>The number of milliseconds</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDateTime</td>
<td>A GlideDateTime object</td>
</tr>
</tbody>
</table>

```javascript
//example script to call the method from a client
Replace MILLISECONDSVALUE with your variable
var ga = new GlideAjax('DateTimeUtils');
```
ga.addParam('sysparm_name','msToGlideDateTime');
ga.addParam('sysparm_value', MILLISECONDSVALUE);
ga.getXMLWait();
var newGDT = ga.getAnswer();

### DurationCalculator

Provides methods for calculating durations and due dates.

### DurationCalculator - calcDuration(Number seconds)

Calculates an end date and time based on a specified start datetime and duration (seconds).

Upon completion, **this.endDateTime**, **this.seconds**, and **this.totalSeconds** properties are set to indicate the results of the calculation. Prior to calling this method, you must call **setStartDateTime()** with the start time to use in the computation.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>seconds</td>
<td>Number</td>
<td>Number of seconds to add to the <strong>startDateTime</strong> to compute the other values.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>False if input value is not a number.</td>
</tr>
</tbody>
</table>

```
var gdt = new GlideDateTime("2012-05-01 00:00:00");
dc.setStartDateTime(gdt);
if(!dc.calcDuration(2*24*3600)){ // 2 days
    gs.log("*** Error calculating duration");
    return;
}
```

### DurationCalculator - calcRelativeDueDate(GlideDateTime start, Number days, String endTime)

Calculates the due date starting at **start** and adding **days** using the schedule and time zone.

Called from relative duration definitions, initiated by **calcRelativeDuration()**, as **calculator.calcRelativeDueDate(calculator.startDateTime, days)**. When the day that the work is due is found, sets the time to **endTime** of that day. If there are not enough days left in the schedule, uses the last day in the schedule.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>GlideDateTime</td>
<td>Start <strong>datetime</strong> for the computation.</td>
</tr>
<tr>
<td>days</td>
<td>Number</td>
<td>Number of days to add to the start <strong>datetime</strong>.</td>
</tr>
<tr>
<td>endTime</td>
<td>String</td>
<td>Time due of the computed day due in &quot;hh:mm:ss&quot; format, or blank to indicate the end of the work day</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the completion date is within the schedule. False if the completion date falls outside the schedule. Undefined if no schedule was set prior to calling this method.</td>
</tr>
</tbody>
</table>

```javascript
var dc = new DurationCalculator();
dc.calcRelativeDueDate("2012-04-10 08:00:00", 2, "08:00:00")
```

**DurationCalculator - calcRelativeDuration(String relativeDurationID)**

Calculates the duration using the specified relative duration script. Upon completion, the **this.endDateTime** and **this.seconds** properties are set to indicate the results of the calculation.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>relativeDurationID</td>
<td>String</td>
<td>sys_id of relative duration schedule (table cmn_relative_duration).</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>The result of the duration script.</td>
</tr>
</tbody>
</table>

```javascript
var dc = new DurationCalculator();
dc.calcRelativeDuration('08fcd0830a0a1b2600074f56b1ad7cb');
```
**DurationCalculator - calcScheduleDuration(String startTime, String endTime)**

Gets the actual duration between `startTime` and `endTime` within the already-specified schedule and optionally overridden timezone. Sets `this.endDateDateTime` (for completeness), `this.seconds`, and `this.totalSeconds`.

Sets `this.endDateDateTime` (for completeness), `this.seconds`, and `this.totalSeconds`.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTime</td>
<td>String</td>
<td>The start time. If not provided the current value is used (set using <code>setStartDateTime()</code>).</td>
</tr>
<tr>
<td>endTime</td>
<td>String</td>
<td>The end time. If not provided the current value is used (set using <code>setEndDateTime()</code>).</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The schedule duration in seconds (same as calling <code>getSeconds()</code>). Returns 0 if <code>endTime</code> is before <code>startTime</code>.</td>
</tr>
</tbody>
</table>

```javascript
var dc = new DurationCalculator();
gs.print(dc.calcScheduleDuration("2012-04-10 08:00:00","2012-04-14 06:00:00"));
```

*** Script: 338400

**DurationCalculator - calcScheduleDuration(GlideDateTime startTime, GlideDateTime endTime)**

Returns the actual duration between `startTime` and `endTime` within the already-specified schedule and optionally overridden timezone.

Sets `this.endDateDateTime` (for completeness), `this.seconds`, and `this.totalSeconds`.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTime</td>
<td>GlideDateTime</td>
<td>The start time. If not provided the current value is used (set using <code>setStartDateTime()</code>).</td>
</tr>
<tr>
<td>endTime</td>
<td>GlideDateTime</td>
<td>The end time. If not provided the current value is used (set using <code>setEndDateTime()</code>).</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>The schedule duration in seconds (same as calling getSeconds()). Returns 0 if endTime is before startTime.</td>
<td></td>
</tr>
</tbody>
</table>

**DurationCalculator - DurationCalculator()**

Constructor for DurationCalculator class.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var dc = new DurationCalculator();
```

**DurationCalculator - getEndDateTime()**

Gets the **endDateTime** property that was set by calcDuration/calcRelativeDuration, indicating the end date and time for the duration.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDateTime</td>
<td>The end datetime.</td>
</tr>
</tbody>
</table>

```javascript
var dc = new DurationCalculator();
dc.calcDuration(52);
gs.print(dc.getEndDateTime());
```

2012-04-17 20:57:27

**DurationCalculator - getSeconds()**

Gets the **this.seconds** property that was set by calcDuration/calcRelativeDuration, indicating the total number of seconds of work to be performed for the duration. This is the...
total work time, not the total time between start and end times and may be used to determine percentages of the work time.

This is the total work time, not the total time between start and end times and may be used to determine percentages of the work time.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The total work time, in seconds.</td>
</tr>
</tbody>
</table>

```javascript
var dc = new DurationCalculator();
dc.calcDuration(52);
gs.print(dc.getSeconds());
```

*** Script: 52

**DurationCalculator - getTotalSeconds()**

Gets the `totalSeconds` value that was set by a call to `calculate(record)`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The total number of seconds.</td>
</tr>
</tbody>
</table>

```javascript
var dc = new DurationCalculator();
dc.calcDuration(52);
gs.print(dc.getTotalSeconds());
```

*** Script: 52

**DurationCalculator - isAfter(GlideDateTime dt, String tm)**

Checks if a time occurs after a specified time of day.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dt</td>
<td>GlideDateTime</td>
<td>The Datetime to compare to.</td>
</tr>
<tr>
<td>tm</td>
<td>String</td>
<td>The test datetime.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if tm is after dt.</td>
</tr>
</tbody>
</table>

```javascript
var dc = new DurationCalculator();
gs.print(dc.isAfter("2012-04-10 08:00:00", "07:00:00"));
```

*** Script: true

---

### DurationCalculator - setSchedule(String schedId, String timezone)

Sets the schedule and time zone to use for calculating the due date.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>schedId</td>
<td>String</td>
<td>The schedule sys_id (table cmn_schedule).</td>
</tr>
<tr>
<td>timezone</td>
<td>String</td>
<td>(Optional) Time zone string value.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var dc = new DurationCalculator();
dc.setSchedule('08fcd0830a0a0b2600079f56b1ad9ae', 'Los Angeles');
```

### DurationCalculator - setStartDateTime(String start)

Sets the start datetime for the duration calculations.

---

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>String</td>
<td>(Optional) The start time in GMT for subsequent calculations. If not present then the current <code>datetime</code> is used.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var dc = new DurationCalculator();
dc.setStartDateTime("2012-04-10 08:00:00")
```

**DurationCalculator - setStartDateTime(GlideDateTime description, Number amount)**

Sets the start `datetime` for the duration calculations.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>GlideDateTime</td>
<td>(Optional) The start time in GMT for subsequent calculations. If not present then the current <code>datetime</code> is used.</td>
</tr>
<tr>
<td>amount</td>
<td>Number</td>
<td>amount</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var dc = new DurationCalculator();
dc.calcDuration(52);
gs.print(dc.getEndDateTime());
```

**DurationCalculator - setTimeZone(String timezone)**

Sets the time zone to use for calculating the due date.

```javascript
```
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timezone</td>
<td>String</td>
<td>Value of the time zone.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var dc = new DurationCalculator();
dc.setTimeZone("Los Angeles");

**ExpenseAllocation**

The ExpenseAllocation API is included with the Cost Management Plugin as a script include record. It is used by various cost management processes and can also be used for generating custom expense allocation records (fm_expense_allocation) from scripted expense allocation rules.

**ExpenseAllocation - createAllocation(GlideRecord target, Number amount)**

Creates an expense allocation (fm_expense_allocation) record referencing the parameters provided during instantiation and this method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>GlideRecord</td>
<td>GlideRecord target of the allocation, for example a cost center record to allocate an expense to Decimal amount - the amount of the allocation.</td>
</tr>
<tr>
<td>amount</td>
<td>Number</td>
<td>The amount of the allocation.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the expense allocation was successfully created.</td>
</tr>
</tbody>
</table>

var allocation=new ExpenseAllocation(expenseGlideRecord, ruleGlideRecord);
allocation.createAllocation(costCenterGlideRecord, 2345.67);
**ExpenseAllocation - ExpenseAllocation(GlideRecord expense, GlideRecord rule)**

Called when you create a new ExpenseAllocation object.

This is not needed if scripting advanced allocation rules. This object is already available as the allocation variable.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expense</td>
<td>GlideRecord</td>
<td>GlideRecord identifying the source of the expense.</td>
</tr>
<tr>
<td>rule</td>
<td>GlideRecord</td>
<td>GlideRecord identifying the rule to use in allocating the expense line.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExpenseAllocation object</td>
<td>The ExpenseAllocation object just created.</td>
</tr>
</tbody>
</table>

```javascript
var allocation=new ExpenseAllocation(expenseGlideRecord,
ruleGlideRecord);
```

**ExpenseLine**

The ExpenseLine API is included with the Cost Management Plugin as a script include record. It is used by various cost management processes and can also be used for generating expense line (fm_expense_line) records from your own server-side scripts.

**ExpenseLine - createExpense()**

Creates a new expense line record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the expense line was successfully created.</td>
</tr>
</tbody>
</table>
//get some random CI to be used as an expense source
var ci = new GlideRecord("cmdb_ci_server");
    ci.query();
    ci.next();

//create expense line
var exp = new ExpenseLine(ci, 234.56, "Test expense line");
    exp.setSummaryType("run_business");
    var success = exp.createExpense();

**ExpenseLine - ExpenseLine (GlideRecord source, Number amount, String description)**

Constructor for ExpenseLine object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>GlideRecord</td>
<td>GlideRecord identifying the source of the expense</td>
</tr>
<tr>
<td>amount</td>
<td>Number</td>
<td>Decimal number identifying the amount of the expense</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>(Optional) Description of the expense</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExpenseLine object</td>
<td>The ExpenseLine object just instantiated.</td>
</tr>
</tbody>
</table>

//get some random CI to be used as an expense source
var ci = new GlideRecord("cmdb_ci_server");
    ci.query();
    ci.next();

//create expense line
var exp = new ExpenseLine(ci, 234.56, "Test expense line");

**ExpenseLine - processCIParents()**

Used internally by the createExpense method to process CI relationships when the expense source is a cmdb_ci record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ExpenseLine - setCostSource(GlideRecord costSource)**

Identifies the source rate card or distribution cost that was the source of expense line generation. This is not the source (CI, task) of the expense.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>costSource</td>
<td>GlideRecord</td>
<td>GlideRecord of CI rate card cost, distribution cost, or task rate card. This is generally only used for system-generated expense lines.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ExpenseLine - setDescription(String description)**

Defines the description of an expense.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>String</td>
<td>Description of expense.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ExpenseLine - setParent(GlideRecord expense)**

Sets the parent field on the expense line. This is generally only used by the system when generating indirect expenses such as business service aggregated expenses.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expense</td>
<td>GlideRecord</td>
<td>Parent expense line record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ExpenseLine - setRecurring(Boolean recurring)**

Flags the expense as recurring by setting the recurring field to true. Expense lines are set to false by default so there is no need to call `setRecurring(false)`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>recurring</td>
<td>Boolean</td>
<td>Set to true to identify expense line as a recurring expense.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ExpenseLine - setSummaryType(String summaryType)**

Sets a value for the expense line summary_type field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>summaryType</td>
<td>String</td>
<td>Typically you would set this to a value already specified in the expense line summary_type field choice list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
//get some random CI to be used as an expense sourcevar ci = new GlideRecord("cmdb_ci_server");
```
ci.query();
ci.next();

//create expense line
var exp = new ExpenseLine(ci, 234.56, "Test expense line");
exp.setSummaryType("run_business");

ExtractTermsFromAttachment

Extracts terms from an attachment.
This class is called with the ScriptedExtractor object, SysAttachmentInputStream, the sys_id for the attachment, and the extension for the attachment. The getTerms() method is called to extract the terms from the attachment that should be indexed. The getTerms() method should just return a string that contains the terms. If you prefer to input a file rather than an inputStream, call extractor.getFile() to get the File object containing the attachment.

FormInfoHeader

FormInfoHeader allows you to add an HTML message as a form info message.
The addMessage function is commonly used in record producers.
The FormInfoHeader class is available to server-side script.

FormInfoHeader - addMessage(String message)

Adds an HTML message to the form header, where form info messages are displayed.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>A message that may include HTML tags.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

gs.include("FormInfoHeader");
var fi = new FormInfoHeader();
fi.addMessage('This incident was opened on your behalf<br/>
The IT department will contact you for further information
or when the incident is resolved');

GenericUPSAlarmsReconciler

Concrete reconciler for generic UPS alarms.
Use this API for SNMP-related discovery.
**GenericUPSAlarmsReconciler - getReconciliationField()**

Returns the reconciliation field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The reconciliation field</td>
</tr>
</tbody>
</table>

**GenericUPSAlarmsReconciler - getReconciliationKey()**

Returns the reconciliation key.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The reconciliation key</td>
</tr>
</tbody>
</table>

**GenericUPSAlarmsReconciler - hasChanged()**

Determines if the generic UPS Bypass information has changed.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the information has changed; otherwise, false.</td>
</tr>
</tbody>
</table>

**GenericUPSAlarmsReconciler - readDatabaseFields()**

Reads the database fields.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GenericUPSAlarmsReconciler - readDiscovered()**

Reads the discovered information.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GenericUPSAlarmsReconciler - setDatabaseFields()**

Sets the database fields.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GenericUPSReconciler**

Concrete reconciler for generic UPS bypasses.

Use this API for SNMP-related discovery.

**GenericUPSReconciler - getReconciliationField()**

Returns the reconciliation field.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The reconciliation field</td>
</tr>
</tbody>
</table>

**GenericUPSReconciler - getReconciliationKey()**

Returns the reconciliation key.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The reconciliation key</td>
</tr>
</tbody>
</table>

**GenericUPSReconciler - hasChanged()**

Determines if the generic UPS Bypass information has changed.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the information has changed; otherwise, false.</td>
</tr>
</tbody>
</table>

**GenericUPSReconciler - readDatabaseFields()**

Reads the database fields.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GenericUPSReconciler - readDiscovered()

Reads the discovered information.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GenericUPSReconciler - setDatabaseFields()

Sets the database fields.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GenericUPSInputsReconciler

Concrete reconciler for generic UPS inputs.

Use this API for SNMP-related discovery.

GenericUPSInputsReconciler - getReconciliationField()

Returns the reconciliation field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### GenericUPSInputsReconciler - getReconciliationKey()

Returns the reconciliation key.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The reconciliation field</td>
</tr>
</tbody>
</table>

### GenericUPSInputsReconciler - hasChanged()

Determines if the generic UPS Bypass information has changed.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the information has changed; otherwise, false.</td>
</tr>
</tbody>
</table>

### GenericUPSInputsReconciler - readDatabaseFields()

Reads the database fields.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
GenericUPSInputsReconciler - readDiscovered()
Reads the discovered information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GenericUPSInputsReconciler - setDatabaseFields()
Sets the database fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GenericUPSOutputsReconciler
Concrete reconciler for generic UPS outputs.
Use this API for SNMP-related discovery.

GenericUPSOutputsReconciler - getReconciliationField()
Returns the reconciliation field.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The reconciliation field</td>
</tr>
</tbody>
</table>
### GenericUPSOutputsReconciler - getReconciliationKey()

Returns the reconciliation key.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### GenericUPSOutputsReconciler - hasChanged()

Determines if the generic UPS bypass information has changed.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the information has changed; otherwise, false.</td>
</tr>
</tbody>
</table>

### GenericUPSOutputsReconciler - readDatabaseFields()

Reads the database fields.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### GenericUPSOutputsReconciler - readDiscovered()

Reads the discovered information.
**GenericUPSOutputsReconciler - setDatabaseFields()**

Sets the database fields.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideAggregate**

GlideAggregate enables you to easily create database aggregation queries.

The GlideAggregate class is an extension of GlideRecord and provides database aggregation (COUNT, SUM, MIN, MAX, AVG) queries. This functionality can be helpful when creating customized reports or in calculations for calculated fields. The GlideAggregate class works only on number fields.

When you use GlideAggregate on currency or price fields, you are working with the reference currency value. Be sure to convert the aggregate values to the user's session currency for display. Because the conversion rate between the currency or price value (displayed value) and its reference currency value (aggregation value) might change, the result may not be what the user expects.

**Note:** When using an on-premise system, the database server time zone must be set to GMT/UTC for this class to work properly.

**GlideAggregate - addAggregate(String agg, String name)**

Adds an aggregate.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agg</td>
<td>String</td>
<td>Name of the aggregate to add, for example, COUNT, MIN, or MAX</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>(Optional) Name of the column to aggregate. Null is the default.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
function doMyBusinessRule(assigned_to, number) {
    var agg = new GlideAggregate('incident');
    agg.addQuery('assigned_to', assigned_to);
    agg.addQuery('category', number);
    agg.addAggregate("COUNT");
    agg.query();
    var answer = 'false';
    if (agg.next()) {
        answer = agg.getAggregate("COUNT");
        if (answer > 0)
            answer = 'true';
        else
            answer = 'false';
    }
    return answer;
}
```

Scoped equivalent

To use the `addAggregate()` method in a scoped application, use the corresponding scoped method: `addAggregate()`.

GlideAggregate - `addEncodedQuery(String query)`

Adds an encoded query to the other queries that may have been set for this aggregate.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>String</td>
<td>An encoded query string to add to the aggregate.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count', 'category');
agg.orderBy('category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
agg.query();
while (agg.next()) {
    var category = agg.category;
    var count = agg.getAggregate('count','category');
    var query = agg.getQuery();
    var agg2 = new GlideAggregate('incident');
    agg2.addAggregate('count','category');
    agg2.orderByAggregate('count', 'category');
    agg2.orderBy('category');
    agg2.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(3)');
    agg2.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(3)');
    agg2.addEncodedQuery(query);
    agg2.query();
    var last = "";
    while (agg2.next()) {
        last = agg2.getAggregate('count','category');
    }
    gs.log(category + " : Last month:" + count + " Previous Month:" + last);
}
```

Scoped equivalent

To use the `addEncodedQuery()` method in a scoped application, use the corresponding scoped method: `addEncodedQuery()`.

**GlideAggregate - addHaving(String name, String operator, String value)**

Adds a "having" element to the aggregate e.g. select category, count(*) from incident group by category HAVING count(*) > 5.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The aggregate to filter on for example, COUNT.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>operator</td>
<td>String</td>
<td>The operator symbol for example &lt;, &gt;, =, !=.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to query on, for example '5'.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideAggregate - addTrend(String fieldName, String timeInterval)**

Adds a trend for a field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The name of the field for which trending should occur.</td>
</tr>
<tr>
<td>timeInterval</td>
<td>String</td>
<td>The time interval for the trend. The following choices are available: year, quarter, date, week, month, dayofweek, hour, value.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var trend = new GlideAggregate('incident');
trend.addTrend ('opened_at','Month');
trend.addAggregate('COUNT');
trend.setGroup(false);
trend.query();
while(trend.next()) {
    gs.print(trend.getValue('timeref') + ': ' +
    trend.getAggregate('COUNT'));
}
```

**Scoped equivalent**

To use the `addTrend(String fieldName, String timeInterval)` method in a scoped application, use the corresponding scoped method: `addTrend(String fieldName, String timeInterval)Scoped GlideAggregate - addTrend(String fieldName, String timeInterval).`
GlideAggregate - `getAggregate(String agg, String name)`

Returns the value of an aggregate from the current record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agg</td>
<td>String</td>
<td>The type of the aggregate, for example, SUM or Count.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the field to get the aggregate from.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the aggregate.</td>
</tr>
</tbody>
</table>

```javascript
function doMyBusinessRule(assigned_to, number) {
  var agg = new GlideAggregate('incident');
  agg.addQuery('assigned_to', assigned_to);
  agg.addQuery('category', number);
  agg.addAggregate("COUNT");
  agg.query();
  var answer = 'false';
  if (agg.next()) {
    answer = agg.getAggregate("COUNT");
    if (answer > 0)
      answer = 'true';
    else
      answer = 'false';
  }
  return answer;
}
```

**Scoped equivalent**

To use the `getAggregate()` method in a scoped application, use the corresponding scoped method: `getAggregate()`.

GlideAggregate - `getQuery()`

Returns the query necessary to return the current aggregate.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### GlideAggregate - getTotal(String agg, String name)

Returns the number of records by summing an aggregate.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agg</td>
<td>String</td>
<td>The aggregate</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the field to aggregate</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The total</td>
</tr>
</tbody>
</table>
### GlideAggregate - `getValue(String name)`

Returns the value of a field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the field.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the field.</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `getValue()` method in a scoped application, use the corresponding scoped method: `getValue()`.

### GlideAggregate - `groupBy(String name)`

Provides the name of a field to use in grouping the aggregates. May be called numerous times to set multiple group fields.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
count.addAggregate('MIN', 'sys_mod_count');
count.addAggregate('MAX', 'sys_mod_count');
count.addAggregate('AVG', 'sys_mod_count');
count.groupBy('category');
count.query();
while (count.next()) {
    var min = count.getAggregate('MIN', 'sys_mod_count');
    var max = count.getAggregate('MAX', 'sys_mod_count');
    var avg = count.getAggregate('AVG', 'sys_mod_count');
    var category = count.category.getDisplayValue();
    gs.log(category + " Update counts: MIN = " + min + " MAX = " + max + " AVG = " + avg);
```
Scoped equivalent

To use the `groupBy()` method in a scoped application, use the corresponding scoped method: `groupBy()`.  

**GlideAggregate - orderBy(String name)**

Provides the name of a field that should be used to order the aggregates. The field will also be added to the group-by list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the field used to order the aggregates.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count', 'category');
agg.orderBy('category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
agg.query();
while (agg.next()) {
    var category = agg.category;
    var count = agg.getAggregate('count','category');
    var query = agg.getQuery();
    var agg2 = new GlideAggregate('incident');
    agg2.addAggregate('count','category');
    agg2.orderByAggregate('count', 'category');
    agg2.orderBy('category');
    agg2.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(3)');
    agg2.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(3)');
    agg2.addEncodedQuery(query);
    agg2.query();
    var last = "";
    while (agg2.next()) {
        last = agg2.getAggregate('count','category');
    }
    gs.log(category + ": Last month:" + count + " Previous Month:" + last);```
**Scoped equivalent**

To use the `orderBy()` method in a scoped application, use the corresponding scoped method: `orderBy()`.

**GlideAggregate - orderByAggregate(String agg, String name)**

Orders the aggregates based on the specified aggregate and field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agg</td>
<td>String</td>
<td>Type of aggregation, for example SUM, COUNT, MIN, MAX.</td>
</tr>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field to aggregate.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count', 'category');
agg.orderBy('category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
agg.query();
while (agg.next()) {
  var category = agg.category;
  var count = agg.getAggregate('count','category');
  var query = agg.getQuery();
  var agg2 = new GlideAggregate('incident');
  agg2.addAggregate('count','category');
  agg2.orderByAggregate('count', 'category');
  agg2.orderBy('category');
  agg2.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(3)');
  agg2.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(3)');
  agg2.addEncodedQuery(query);
  agg2.query();
  var last = "";
  while (agg2.next()) {
    last = agg2.getAggregate('count','category');
    
```
Scoped equivalent

To use the `orderByAggregate()` method in a scoped application, use the corresponding scoped method: `orderByAggregate()`.

GlideAggregate - `query()`

Issues the query and gets the results.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count', 'category');
agg.orderBy('category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
agg.query();
while (agg.next()) {
    var category = agg.category;
    var count = agg.getAggregate('count','category');
    var query = agg.getQuery();
    var agg2 = new GlideAggregate('incident');
    agg2.addAggregate('count','category');
    agg2.orderByAggregate('count', 'category');
    agg2.orderBy('category');
    agg2.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(3)');
    agg2.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(3)');
    agg2.addEncodedQuery(query);
    agg2.query();
    var last = "";
    while (agg2.next()) {
        last = agg2.getAggregate('count','category');
    }
```
gs.log(category + ": Last month:" + count + ": Previous Month:" + last);

Scoped equivalent

To use the `query()` method in a scoped application, use the corresponding scoped method: `query()`.

GlideAggregate - `setGroup(Boolean b)`

Sets whether the results are to be grouped.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>Set to true if grouping is true, otherwise set to false.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var ga = new GlideAggregate('incident');
ga.addAggregate('COUNT', 'category');
ga.setGroup(true);
```

Scoped equivalent

To use the `setGroup()` method in a scoped application, use the corresponding scoped method: `setGroup()`.

GlideAggregate

GlideAggregate enables you to easily create database aggregation queries.

The scoped GlideAggregate class is an extension of GlideRecord and provides database aggregation (COUNT, SUM, MIN, MAX, AVG) queries. This functionality can be helpful when creating customized reports or in calculations for calculated fields. The GlideAggregate class works only on number fields.

When you use GlideAggregate on currency or price fields, you are working with the reference currency value. Be sure to convert the aggregate values to the user’s session currency for display. Because the conversion rate between the currency or price value (displayed value) and its reference `currency` value (aggregation value) might change, the result may not be what the user expects.
Note: When using an on-premise system, the database server time zone must be set to GMT/UTC for this class to work properly.

Scoped GlideAggregate - addAggregate(String agg, String name)

Adds an aggregate.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agg</td>
<td>String</td>
<td>Name of the aggregate to add, for example, COUNT, MIN, or MAX</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>(Optional) Name of the column to aggregate. Null is the default.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if (count.next()) {
    incidents = count.getAggregate('COUNT');
}
//Number of incidents varies depending on the current state
//of the incident table
gs.info('Number of incidents: ' + incidents);
```

Scoped GlideAggregate - addEncodedQuery(String query)

Adds an encoded query to the other queries that may have been set for this aggregate.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>String</td>
<td>An encoded query to add to the aggregate.</td>
</tr>
</tbody>
</table>
//Number of incidents varies depending on the current state
//of the incident table
var count = new GlideAggregate('incident');
count.addEncodedQuery('active=true');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if (count.next())
    incidents = count.getAggregate('COUNT');
gs.info(incidents);

Scoped GlideAggregate - addQuery(String name, String operator, String value)

Adds a query to the aggregate.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The query to add.</td>
</tr>
<tr>
<td>operator</td>
<td>String</td>
<td>The operator for the query.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The list of values to include in the query.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>The query condition.</td>
</tr>
</tbody>
</table>

//Number of incidents varies depending on the current state
//of the incident table
var count = new GlideAggregate('incident');
count.addQuery('active','=',true);
count.addAggregate('COUNT','category');
count.query();
while (count.next()) {
    var category = count.category;
    var categoryCount = count.getAggregate('COUNT','category');
    gs.info("There are currently " + categoryCount + " incidents with a category of " + category);
}

Output:

There are currently 1 incidents with a category of database
There are currently 5 incidents with a category of hardware
There are currently 42 incidents with a category of inquiry
There are currently 4 incidents with a category of network
There are currently 4 incidents with a category of request
There are currently 7 incidents with a category of software

Scoped GlideAggregate - addNotNullQuery(String fieldName)

Adds a not null query to the aggregate.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The name of the field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>The scoped query condition.</td>
</tr>
</tbody>
</table>

var count = new GlideAggregate('incident');
count.addNotNullQuery('short_description');
count.query();   // Issue the query to the database to get all records
while (count.next()) {
    // add code here to process the aggregate
}

Scoped GlideAggregate - addNullQuery(String fieldName)

Adds a null query to the aggregate.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The name of the field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>The scoped query condition.</td>
</tr>
</tbody>
</table>

var count = new GlideAggregate('incident');
count.addNullQuery('short_description');
count.query();   // Issue the query to the database to get all records
while (count.next()) {
    // add code here to process the aggregate
}
Scoped GlideAggregate - addTrend(String fieldName, String timeInterval)

Adds a trend for a field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The name of the field for which trending should occur.</td>
</tr>
<tr>
<td>timeInterval</td>
<td>String</td>
<td>The time interval for the trend. The following choices are available: Year, Quarter, Date, Week, DayOfWeek, Hour, Value.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Scoped GlideAggregate - getAggregate(String agg, String name)

Gets the value of an aggregate from the current record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agg</td>
<td>String</td>
<td>The type of the aggregate, for example, SUM or Count.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the field to get the aggregate from.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the aggregate.</td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if (count.next()) {
    incidents = count.getAggregate('COUNT');
}
//Number of incidents varies depending on the current state
//of the incident table
gs.info('Number of incidents: ' + incidents);
```
Output: Number of incidents: 63

**Scoped GlideAggregate - getAggregateEncodedQuery()**

Gets the query necessary to return the current aggregate.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The encoded query to get the aggregate.</td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
count.addAggregate('MIN', 'sys_mod_count');
count.groupBy('category');
count.query();
while (count.next()) {
    gs.info(count.getAggregateEncodedQuery());
}
```

**Output:**

category=database
category=hardware
category=inquiry
category=network
category=request
category=software

**Scoped GlideAggregate - getEncodedQuery()**

Retrieves the encoded query.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The encoded query.</td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
```
count.addAggregate('MIN', 'sys_mod_count');
count.addAggregate('MAX', 'sys_mod_count');
count.addAggregate('AVG', 'sys_mod_count');
count.groupBy('category');
count.query();
gs.info(count.getEncodedQuery());

Output:

ORDERBYcategory^GROUPBYcategory

Scoped GlideAggregate - getRowCount()
Retrieves the number of rows in the GlideAggregate object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of rows in the GlideAggregate object.</td>
</tr>
</tbody>
</table>

var count = new GlideAggregate('incident');
count.addAggregate('MIN', 'sys_mod_count');
count.addAggregate('MAX', 'sys_mod_count');
count.addAggregate('AVG', 'sys_mod_count');
count.groupBy('category');
count.query();
gs.info(count.getRowCount());
while (count.next()) {
  var min = count.getAggregate('MIN', 'sys_mod_count');
  var max = count.getAggregate('MAX', 'sys_mod_count');
  var avg = count.getAggregate('AVG', 'sys_mod_count');
  var category = count.category.getDisplayValue();
  gs.info(category + " Update counts: MIN = " + min + " MAX = " + max + " AVG = " + avg);
}

Output:

| Database Update counts: MIN = 8 MAX = 48 AVG = 28.0000 |
| Hardware Update counts: MIN = 4 MAX = 14 AVG = 6.6250   |
| Inquiry / Help Update counts: MIN = 0 MAX = 34 AVG = 6.5714 |
| Network Update counts: MIN = 3 MAX = 37 AVG = 18.6000   |
| Request Update counts: MIN = 5 MAX = 39 AVG = 13.4000   |
| Software Update counts: MIN = 4 MAX = 98 AVG = 24.0000   |
Scoped GlideAggregate - getTableName()

Retrieves the table name associated with this GlideAggregate object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The table name.</td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
count.addAggregate('MIN', 'sys_mod_count');
count.addAggregate('MAX', 'sys_mod_count');
count.addAggregate('AVG', 'sys_mod_count');
count.groupBy('category');
count.query();
gs.info(count.getTableName());
```

Scoped GlideAggregate - getValue(String name)

Gets the value of a field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the field.</td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
count.addAggregate('MIN', 'sys_mod_count');
count.addAggregate('MAX', 'sys_mod_count');
count.addAggregate('AVG', 'sys_mod_count');
count.groupBy('category');
count.query();
while (count.next()) {
    var min = count.getAggregate('MIN', 'sys_mod_count');
    var max = count.getAggregate('MAX', 'sys_mod_count');
    var avg = count.getAggregate('AVG', 'sys_mod_count');
    var category = count.category.getDisplayValue();
}
gs.info("Current category is: " + count.getValue('category'));

Output: Current category is: software

**Scoped GlideAggregate - GlideAggregate(String tableName)**

Creates a GlideAggregate object on the specified table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Name of the table.</td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
```

**Scoped GlideAggregate - groupBy(String name)**

Provides the name of a field to use in grouping the aggregates. May be called numerous times to set multiple group fields.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
count.addAggregate('MIN', 'sys_mod_count');
count.addAggregate('MAX', 'sys_mod_count');
count.addAggregate('AVG', 'sys_mod_count');
count.groupBy('category');
count.query();
while (count.next()) {
    var min = count.getAggregate('MIN', 'sys_mod_count');
    var max = count.getAggregate('MAX', 'sys_mod_count');
    var avg = count.getAggregate('AVG', 'sys_mod_count');
    var category = count.category.getDisplayValue();
    gs.info(category + " Update counts: MIN = " + min + " MAX = " + max + " AVG = " + avg);
}
```

Output:

Database Update counts: MIN = 8 MAX = 48 AVG = 28.0000
Hardware Update counts: MIN = 4 MAX = 14 AVG = 6.6250
Scoped GlideAggregate - hasNext()

Determines if there are any more records in the GlideAggregate object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if there are more results in the query set.</td>
</tr>
</tbody>
</table>

```javascript
var agg = new GlideAggregate('incident');
agg.addAggregate('AVG', 'sys_mod_count');
agg.groupBy('category');
agg.query();
while (agg.hasNext()) {
    agg.next();
    var avg = agg.getAggregate('AVG', 'sys_mod_count');
    var category = agg.category.getDisplayValue();
    gs.info(category + ': AVG = ' + avg);
}
```

Output:

Database: AVG = 32.5000
Hardware: AVG = 12.0000
Inquiry / Help: AVG = 7.6667
Network: AVG = 24.0000
Request: AVG = 16.4000
Software: AVG = 27.0833

Scoped GlideAggregate - next()

Moves to the next record in the GlideAggregate.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if there are more records in the query set; otherwise, false.</td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if (count.next()) {
    incidents = count.getAggregate('COUNT');
gs.info(incidents);
}
```

### Scoped GlideAggregate - orderBy(String name)

Orders the aggregates using the value of the specified field. The field will also be added to the group-by list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the field to order the aggregates by.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var agg = new GlideAggregate('incident');
agg.addAggregate('count', 'category');
agg.orderBy('category');
gs.query();
while (agg.next()) {
    var category = agg.category;
    var count = agg.getAggregate('count', 'category');
    var agg2 = new GlideAggregate('incident');
    agg2.addAggregate('count', 'category');
    agg2.orderBy('category');
gs.info(category + ': Current number of incidents:' + count);
}
```

**Output:**

```
database: Current number of incidents:2
hardware: Current number of incidents:8
inquiry: Current number of incidents:28
network: Current number of incidents:5
```
request: Current number of incidents:5
software: Current number of incidents:11

Scoped GlideAggregate - orderByAggregate(String agg, String fieldName)

Orders the aggregates based on the specified aggregate and field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agg</td>
<td>String</td>
<td>Type of aggregation.</td>
</tr>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field to aggregate.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
ga.addAggregate('COUNT', 'category');
ga.orderByAggregate('count', 'category');
ga.query();
while(ga.next()) {
    gs.info('Category ' + ga.category + ' ' + ga.getAggregate('COUNT', 'category'));
}
```

Output:

Category inquiry 18
Category software 11
Category hardware 7
Category network 5
Category request 5
Category 4
Category database 2

Scoped GlideAggregate - orderByDesc(String name)

Sorts the aggregates in descending order based on the specified field. The field will also be added to the group-by list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
</tbody>
</table>
ServiceNow    Kingston    Now Platform Custom Business Applications

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var agg = new GlideAggregate('incident');
agg.addAggregate('count', 'category');
agg.orderByDesc('category');
agg.query();
while (agg.next()) {
    var category = agg.category;
    var count = agg.getAggregate('count', 'category');
    var agg2 = new GlideAggregate('incident');
    agg2.addAggregate('count', 'category');
    agg2.orderBy('category');
    gs.info(category + ": Current number of incidents:" + count);
}
```

Output:

```
software: Current number of incidents:11
request: Current number of incidents:5
network: Current number of incidents:5
inquiry: Current number of incidents:28
hardware: Current number of incidents:8
database: Current number of incidents:2
```

Scoped GlideAggregate - query()
Issues the query and gets the results.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if (count.next()) {
    incidents = count.getAggregate('COUNT');
}
gs.info('Number of incidents: ' + incidents);
```
**Scoped GlideAggregate - setGroup(Boolean b)**

Sets whether the results are to be grouped.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>When true the results are grouped.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var ga = new GlideAggregate('incident');
ga.addAggregate('COUNT', 'category');

ga.setGroup(true);
ga.groupBy("category");

ga.query();

while(ga.next()) {
    gs.info('Category ' + ga.category + ' ' + ga.getAggregate('COUNT', 'category'));
}
```

**Output:**

- Category database 2
- Category hardware 7
- Category inquiry 18
- Category network 5
- Category request 5
- Category software 11

**GlideAjax**

The GlideAjax class enables a client script to call server-side code in a script include.

To use GlideAjax in a client script, follow these general steps.

1. Create a GlideAjax instance by calling the GlideAjax constructor. As the argument to the constructor, specify the name of the script include class that contains the method you want to call.
2. Call the addParam method with the sysparm_name parameter and the name of the script-include method you want to call.
3. (Optional) Call the addParam method one or more times to provide the script-include code with other parameters it needs.
4. Execute the server-side code by calling getXML().
Note: `getXML()` is the preferred method for executing the code, because it is asynchronous and does not hold up the execution of other client code. Another method, `getXMLWait()`, is also available but is not recommended. Using `getXMLWait()` ensures the order of execution, but can cause the application to seem unresponsive, significantly degrading the user experience of any application that uses it. `getXMLWait()` is not available to scoped applications.

```javascript
var ga = new GlideAjax('HelloWorld'); // HelloWorld is the script include class
ga.addParam('sysparm_name','helloWorld'); // helloWorld is the script include method
ga.addParam('sysparm_user_name','Bob'); // Set parameter sysparm_user_name to 'Bob'
ga.getXML('HelloWorldParse'); /* Call HelloWorld.helloWorld() with the parameter sysparm_user_name set to 'Bob' and use the callback function HelloWorldParse() to return the result when ready */

// the callback function for returning the result from the server-side code
function HelloWorldParse(response) {
  var answer = 
  response.responseXML.documentElement.getAttribute("answer");
  alert(answer);
}
```

**GlideAjax - getAnswer()**

Retrieves the results from a server-side method called from the client via `getXMLWait()`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| void   | The result returned by the server-side method previously called with `getXMLWait()`.

**GlideAjax - addParam(String parm_name, String parm_value)**

Specifies a parameter name and value to be passed to the server-side function associated with this GlideAjax object.

You can execute `addParam` multiple times with different parameters and values.

**Note:** The first call to `addParam` should be with the parameter `sysparm_name` and the name of the server-side method you want to call. The server-side code does not execute until the client script calls `getXML()`.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parm_name</td>
<td>String</td>
<td>The name of the parameter to pass. (The name must begin with the sysparm_.)</td>
</tr>
<tr>
<td>parm_value</td>
<td>String</td>
<td>The value to assign to parm_name.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideAjax - getXML(Function callback)**

Sends the server a request to execute the method and parameters associated with this GlideAjax object.

The server processes the request asynchronously and -- when ready -- returns the results via the function specified as the callback_function.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>Function</td>
<td>The name of the callback function to process the results returned by the server.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var comments = gel("dialog_comments").value;
var ga = new GlideAjax('validateComments'); //Call script include to escape text
ga.addParam('sysparm_name', 'validateComments');
ga.addParam('sysparm_comments', comments);
ga.getXML(callback);
return false;

function callback(response) {
  var comments = response.responseXML.documentElement.getAttribute("answer");
  comments = trim(comments);
  if (comments == "") {
    //If comments are empty, alert the user and stop submission alert("Please enter your comments before submitting.");
  }
```
} else {
    // If there are comments, close the dialog window and submit them
    GlideDialogWindow.get().destroy(); // Close the dialog window
    g_form.setValue("comments", comments); // Set the "Comments" field with comments in the dialog
}

GlideAjax - getXMLAnswer(Function callback)

Call the processor asynchronously and get the answer element of the response in XML format.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>Function</td>
<td>The callback function. The function receives the answer element of the response in XML format as an argument.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

function updateAttachmentCount(sysid) {
    var ga = new GlideAjax('AttachmentAjax');
    ga.addParam('sysparm_type', 'attachmentCount');
    ga.addParam('sysparm_value', sysid);
    ga.getXMLAnswer(numberOfAttachments, null, sysid); // callback: numberOfAttachments
}

function numberOfAttachments(answer, sysid) {
    // we want to know there are 5 attachments, not 5.0 attachments
    var number = parseInt(answer);
    var buttons = $$('%.attachmentNumber_' + sysid);
    if (buttons[0] == undefined)
        $('header_attachment_list_label').down().innerHTML = number;
    else {
        for (var i = 0; i < buttons.length; i++) {
            buttons[i].innerHTML = number;
        }
    }
}

GlideAjax - getXMLWait()

Sends the server a request to execute the method and parameters associated with this GlideAjax object.

The server processes the request synchronously and will not process further requests from the client until finished. To retrieve the results, the client must call getAnswer(). Using getXMLWait()
ensures the order of execution, but can cause the application to seem unresponsive, significantly degrading the user experience of any application that uses it. We recommend using `getXML()` instead.

**Note:** `getXMLWait()` is not available to scoped applications.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var ga = new GlideAjax('HelloWorld');
ga.addParam('sysparm_name','helloWorld');
ga.addParam('sysparm_user_name','Bob');
ga.getXMLWait();
alert(ga.getAnswer());
```

**Scoped equivalent**

`getXMLWait()` is not available to scoped applications. Instead use the `getXML()` method.

**GlideAjax - GlideAjax(String class_name)**

Constructor for GlideAjax.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class_name</td>
<td>String</td>
<td>The name of the server-side class that contains the method you want to execute.</td>
</tr>
</tbody>
</table>

**GlideAjaxV3**

The GlideAjaxV3 API provides the ability to asynchronously execute server-side scripts from a client-side script.

The GlideAjaxV3 API can be used in client-side scripts using ListV2 and ListV3 APIs.

**GlideAjaxV3 - addParam(String name, String value)**

Set a name-value pair to be sent to the processor.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the parameter. This usually has the prefix <code>sysparm_</code>.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The parameter value.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideAjaxV3 - getJSON(Function callback)**

Call the processor asynchronously and get the answer element of the response in JSON format.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>Function</td>
<td>The callback function. The function receives the answer element of the response as a JSON object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideAjaxV3 - getParam(String name)**

Returns the value of the specified parameter.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the parameter to return.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The specified parameter's value.</td>
</tr>
</tbody>
</table>

**GlideAjaxV3 - getParams()**

Returns the name-value pairs for the request.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The request's name-value pairs.</td>
</tr>
</tbody>
</table>

**GlideAjaxV3 - getProcessor()**

Returns the server-side script that the request is going to use. This returns the `sysparm_processor` parameter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the request's <code>sysparm_processor</code> parameter.</td>
</tr>
</tbody>
</table>

**GlideAjaxV3 - getURL()**

Returns the target URL.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The URL where the Ajax request will be sent.</td>
</tr>
</tbody>
</table>

**GlideAjaxV3 - getXML(Function callback)**

Call the processor asynchronously and get the response in XML format.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>Function</td>
<td>The callback function. The function receives the response as an argument.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideAjaxV3 - getXMLAnswer(Function callback)**

Call the processor asynchronously and get the answer element of the response in XML format.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>Function</td>
<td>The callback function. The function receives the answer element of the response in XML format as an argument.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
function autofillPhoneNumber(sysid) {
  var ga = new GlideAjax('x_abc_myscope.AjaxUtils');
  ga.addParam('sysparm_type', 'getPhoneNumberForUser');
  ga.addParam('sysparm_user', sysid);
  ga.getXMLAnswer(function(answer) {
    g_form.setValue('phone_number', answer);
  });
}
```

**GlideAjaxV3 - GlideAjax(String processor)**

Creates an instance of the GlideAjaxV3 class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>processor</td>
<td>String</td>
<td>The name of the processor (server-side script) to call.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

_GlideAjaxV3 - GlideAjax(String processor, String targetURL)_

Creates an instance of the GlideAjaxV3 class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>processor</td>
<td>String</td>
<td>The name of the processor (server-side script) to call.</td>
</tr>
<tr>
<td>targetURL</td>
<td>String</td>
<td>(Optional) Override the xmlhttp processor url. If this parameter is not specified, the default is xmlhttp.do.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

_GlideAjaxV3 - setErrorCallback(Function callback)_

Sets a callback function to be called if the Ajax request fails.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>Function</td>
<td>The function to be called if the Ajax request fails. The callback function has one parameter, the XMLHttpRequest object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

_GlideAjaxV3 - setProcessor(String serverScript)_

Sets the request's server-side script. The server-side script is also called the processor. This sets the ‘sysparm_processor’ parameter.
**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverScript</td>
<td>String</td>
<td>The server-side script (processor) to receive the request.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideappCalculationHelper**

The GlideappCalculationHelper API is a scriptable object that provides methods that add items to an existing request or request the recalculation of the price of a request.

The methods for this API are used in global server-side scripts (script includes, business rules, etc.). There is currently no support for scoped applications.

**GlideappCalculationHelper - addItemToExistingRequest(String requestId, String catalogID, String quantity)**

Adds one or more of a specified catalog item to an existing request.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestId</td>
<td>String</td>
<td>sys_id of the request to which to add the catalog item(s)</td>
</tr>
<tr>
<td>catalogID</td>
<td>String</td>
<td>sys_id of the catalog item to add to the request</td>
</tr>
<tr>
<td>quantity</td>
<td>String</td>
<td>Number of the specified catalog item to add to the request</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var catItemId = "04b7e94b4f7b4200086eeed18110c7fd";
var requestId = "6eed229047801200e0ef563dbb9a71c2";
var helper = new GlideappCalculationHelper()
helper.addItemToExistingRequest(requestId, catItemId, "1");
```
GlideappCalculationHelper - rebalanceRequest(String requestId)
Recalculates the price of all of the items in a specified request.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestId</td>
<td>String</td>
<td>sys_id of the request for which to recalculate the price</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var catItemId = "04b7e94b4f7b4200086eed18110c7fd";
var requestId = "6eed229047801200e0ef563dbb9a71c2";
var helper = new GlideappCalculationHelper();
//Add an item to the request
helper.addItemToExistingRequest(requestId, catItemId, "1");
//Re-calculate the price of the request after adding the item
helper.rebalanceRequest(requestId);
```

GlideCertificateEncryption

APIs available for encrypting certificates.
Use these methods to generate a hash for the certificate, sign data using a private key, and generate a message authentication code.

GlideCertificateEncryption - generateMac(String key, String algorithm, String data)
Generates the Message Authentication Code (MAC), which is used to authenticate a message.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>String</td>
<td>Key to use to sign the message. Must be Base64 encoded.</td>
</tr>
<tr>
<td>algorithm</td>
<td>String</td>
<td>Algorithm to use to generate the MAC: HmacSHA256, HmacSHA1, HmacMD5, and so on.</td>
</tr>
<tr>
<td>data</td>
<td>String</td>
<td>Data to process.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>MAC in base64 format.</td>
</tr>
</tbody>
</table>

Example

```javascript
var mac = new GlideCertificateEncryption;
var key = "sample_key";
key = GlideStringUtil.base64Encode(key);
mac.generateMac(key, "HmacSHA256", "sample_data");
```

Scoped equivalent

To use the `generateMac()` method in a scoped application, use the corresponding scoped method: `generateMac().`

GlideCertificateEncryption - `getThumbPrint(String certificateID, String algorithm)`

Generates a hash (SHA-1, SHA-256, and so on) for the certificate from Trust Store Cert.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificateID</td>
<td>String</td>
<td>sys_id of the certificate record in the <code>X.509 Certificate (sys_certificate)</code> table.</td>
</tr>
<tr>
<td>algorithm</td>
<td>String</td>
<td>SHA-1, SHA-256, and so on</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Thumbprint in base64 format.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `getThumbPrint()` method in a scoped application, use the corresponding scoped method: `getThumbPrint().`

GlideCertificateEncryption - `getThumbPrintFromKeystore(String certificateID, String alias, String algorithm)`

Generates a hash (SHA-1, SHA-256, and so on) for the certificate from the key store entry.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificateID</td>
<td>String</td>
<td>sys_id of the certificate record in the X.509 Certificate (sys_certificate) table.</td>
</tr>
<tr>
<td>alias</td>
<td>String</td>
<td>Alias name for the certificate.</td>
</tr>
<tr>
<td>algorithm</td>
<td>String</td>
<td>SHA-1, SHA-256, and so on.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Thumbprint in base64 format.</td>
</tr>
</tbody>
</table>

### Scoped equivalent

To use the `getThumbPrintFromKeystore()` method in a scoped application, use the corresponding scoped method: `getThumbPrintFromKeystore()`.


Instantiates a GlideCertificateEncryption object.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Scoped equivalent

To use the `GlideCertificateEncryption()` constructor in a scoped application, use the corresponding scoped constructor: `CertificateEncryption()`.

**GlideCertificateEncryption - sign(String certificateID, String alias, String aliaspassword, String algorithm, String datatosign)**

Signs the data using the private key and the given algorithm.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificateID</td>
<td>String</td>
<td>sys_id of the certificate record in the X.509 Certificate (sys_certificate) table.</td>
</tr>
<tr>
<td>alias</td>
<td>String</td>
<td>Private key name.</td>
</tr>
<tr>
<td>aliaspassword</td>
<td>String</td>
<td>Password for the private key.</td>
</tr>
</tbody>
</table>
### Name | Type | Description
--- | --- | ---
datatosign | String | Data to sign.
algorithm | String | SHA-1, SHA-256, and so on.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Signed data in base64 format.</td>
</tr>
</tbody>
</table>

#### Scoped equivalent

To use the `sign()` method in a scoped application, use the corresponding scoped method: `sign()`.  

```javascript
var gce = new GlideCertificateEncryption;
gce.sign("recordID", "alias", "password", "SHA-1", "sign this data");
```

#### GlideConversation

The GlideConversation API provides access to information in a Connect message. GlideConversation properties are accessed through a global object (conversation) that is available only in Connect action conditions and scripts. Connect conversations are stored on the Live Group Profile (live_group_profile) table.

#### GlideConversation - description

The conversation's description.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
description | String | The conversation's description. |

```javascript
var c= conversation.description;
```

#### GlideConversation - document

The sysID of the document associated with the conversation. This field is set for feed conversations, and contains the sysID of the record being discussed.
<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>document</td>
<td>GlideRecord</td>
<td>The document associated with the conversation.</td>
</tr>
</tbody>
</table>

**GlideConversation - name**

The name of the conversation.

```
var c = conversation.name;
```

**GlideConversation - queueEntry**

The queue entry associated with the conversation.

The queue entry is a reference field pointing to the chat_queue_entry table. This field is only available on support conversations. The queue chat entry is used to track the state of the conversation. This property is not changed if an incident record is created from the support conversation using the `newRecord()` method.

```
var c = conversation.queueEntry;
```

**GlideConversation - sys_id**

The sys_id of the conversation.

```
var c = conversation.sys_id;
```
GlideConversation - table
The name of the table containing the record being discussed.
This field is set for feed conversations, and is set to the table holding the record.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>The name of the table holding the record being discussed.</td>
</tr>
</tbody>
</table>

```
var c = conversation.table;
```

GlideConversation - type
The conversation type.
This is not the message type.
Connect supports the following types.
- peer - direct conversations, which consist of messages between two users.
- connect - group conversations, which consist of messages between more than two users.
- support - Connect Support conversations.
- feed - record conversations, which consist of messages that correspond to comments and work notes on a specific record.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>The conversation type.</td>
</tr>
</tbody>
</table>

```
var c = conversation.type;
```

GlideDate
The scoped GlideDate class provides methods for performing operations on GlideDate objects, such as instantiating GlideDate objects or working with GlideDate fields.

Scoped GlideDate - getByFormat(String format)
Gets the date in the specified date format.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>format</td>
<td>String</td>
<td>the desired date format</td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>the date in the specified format</td>
</tr>
</tbody>
</table>

```javascript
var gd = new GlideDate();
gd.setValue('2015-01-01');
gs.info(gd.getByFormat("dd-MM-yyyy"));
```

Output: 01-01-2015

#### Scoped GlideDate - getDayOfMonthNoTZ()

Gets the day of the month stored by the GlideDate object, expressed in the UTC time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of the month in the UTC time zone, from 1 to 31.</td>
</tr>
</tbody>
</table>

```javascript
//Today's date is 2016-05-13
var gd = new GlideDate();
gs.info(gd.getDayOfMonthNoTZ());
```

Output: 13

#### Scoped GlideDate - getDisplayValue()

Gets the date in the current user's display format and time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The date in the user’s format and time zone. Keep in mind when designing business rules or script includes that this method may return values in different formats for different users.</td>
</tr>
</tbody>
</table>

```javascript
var gd = new GlideDate();
gd.setValue('2015-01-01');
gs.info(gd.getDisplayValue());
```

Output: 2015-01-01

**Scoped GlideDate - getDisplayValueInternal()**

Gets the display value in the internal format (yyyy-MM-dd).

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The date values for the GlideDate object in the current user’s time zone and the internal time format of yyyy-MM-dd.</td>
</tr>
</tbody>
</table>

```javascript
var gd = new GlideDate();
gs.info(gd.getDisplayValueInternal());
```

Output: 2014-10-22

**Scoped GlideDate - getMonthNoTZ()**

Gets the month stored by the GlideDate object, expressed in the UTC time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The numerical value of the month from 1 to 12.</td>
</tr>
</tbody>
</table>

//Today's date is 2016-05-13
var gd = new GlideDate();
gs.info(gd.getMonthNoTZ());

Output: 5

Scoped GlideDate - getValue()

Gets the date value stored in the database by the GlideDate object in the internal format, yyyy-MM-dd, and the system time zone, UTC by default.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The date value in the internal format and system time zone.</td>
</tr>
</tbody>
</table>

var gd = new GlideDate();
gd.setValue('2015-01-01');
gs.info(gd.getValue());

Output: 2015-01-01

Scoped GlideDate - getYearNoTZ()

Gets the year stored by the GlideDate object, expressed in the UTC time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The numerical value of the year.</td>
</tr>
</tbody>
</table>

//Today's date is 2016-05-13
var gd = new GlideDate();
gs.info(gd.getYearNoTZ());

Output: 2016

Scoped GlideDate - GlideDate()

Creates a GlideDate object with the current date time.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scoped GlideDate - setDisplayValue(String asDisplayed)

Sets a date value using the current user's display format and time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>asDisplayed</td>
<td>String</td>
<td>The date in the current user's display format and time zone. The parameter must be formatted using the current user's preferred display format, such as yyyy-MM-dd.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var gd = new GlideDate();
gd.setDisplayValue("2011-01-01");
gs.info(gd.getValue());

Output: 2011-01-01

Scoped GlideDate - setValue(String o)

Sets the date of the GlideDate object.
**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>String</td>
<td>The date and time to use.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gd = new GlideDate();
gd.setValue('2015-01-01');
gs.info(gd.getValue());
```

Output: 2015-01-01

**Scoped GlideDate - subtract(GlideDate start, GlideDate end)**

Gets the duration difference between two GlideDate values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>GlideDate</td>
<td>The start value.</td>
</tr>
<tr>
<td>end</td>
<td>GlideDate</td>
<td>The end value.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDuration</td>
<td>The duration between the two values.</td>
</tr>
</tbody>
</table>

```javascript
var sgd1 = new GlideDate();
sgd1.setDisplayValue('2014-07-18');
var sgd2 = new GlideDate();
sgd2.setDisplayValue('2014-07-19');

duration= GlideDate.subtract(sgd1, sgd2);
gs.info(duration.getDisplayValue());
```

Output: 1 Day

**GlideDateTime**

The GlideDateTime class provides methods for performing operations on GlideDateTime objects, such as instantiating GlideDateTime objects or working with glide_date_time fields.
Use the GlideDateTime methods to perform date-time operations, such as instantiating a GlideDateTime object, performing date-time calculations, formatting a date-time, or converting between date-time formats.

**GlideDateTime - add(Number milliseconds)**

Adds a specified number of milliseconds to the GlideDateTime object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>Number</td>
<td>The number of milliseconds to add</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getNumericValue());
gdt.add(10);
gs.print(gdt.getNumericValue());
```

Output:

```
1314777600000
1314777600010
```

**Scoped equivalent**

To use the `add()` method in a scoped application, use the corresponding scoped method: `add0`.

**GlideDateTime - add(GlideTime time)**

Adds a GlideTime object to the current GlideDateTime object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>GlideTime</td>
<td>The time to add.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
var gtime1 = new GlideTime();
gtime1.setValue("00:00:20");
gdt.add(gtime1);
gs.print(gdt.getTime());
```

Output: 1970-01-01 08:00:20

**Scoped equivalent**

To use the `add()` method in a scoped application, use the corresponding scoped method: `add()`.

**GlideDateTime - addDays(Number days)**

Adds a specified number of days to the current GlideDateTime object. A negative parameter subtracts days.

Use `addDaysLocalTime()` and `addDaysUTC()` instead of this method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>Number</td>
<td>The number of days to add. Use a negative number to subtract.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addDays(-1);
gs.print(gdt.getDate());
```

Output: 2011-08-30

**GlideDateTime - addDaysLocalTime(Number days)**

Adds a specified number of days to the current GlideDateTime object. A negative parameter subtracts days.

The method determines the local date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts days using the local date and time values.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>Number</td>
<td>The number of days to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addDaysLocalTime(-1);
gs.print(gdt.getLocalDate());
```

Output: 2011-08-30

Scoped equivalent

To use the `addDaysLocalTime()` method in a scoped application, use the corresponding scoped method: `addDaysLocalTime()`.

**GlideDateTime - addDaysUTC(Number days)**

Adds a specified number of days to the current GlideDateTime object. A negative parameter subtracts days.

The method determines the UTC date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts days using the UTC date and time values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>Number</td>
<td>The number of days to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addDaysUTC(-1);
gs.print(gdt.getDate());
```
Output: 2011-08-30

Scoped equivalent

To use the `addDaysUTC()` method in a scoped application, use the corresponding scoped method: `addDaysUTC0`.

GlideDateTime - `addSeconds(Number seconds)`

Adds a specified number of seconds to the GlideDateTime object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>seconds</td>
<td>Number</td>
<td>The number of seconds to add</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-07 08:00:00");
gdt.addSeconds(1000);
gs.print(gdt.getValue());
```

Output: 2011-12-07 08:16:40

Scoped equivalent

To use the `addSeconds()` method in a scoped application, use the corresponding scoped method: `addSeconds0`.

GlideDateTime - `addWeeks(Number weeks)`

Adds a specified number of weeks to the current GlideDateTime object. A negative parameter subtracts weeks.

Use `addWeeksLocalTime()` and `addWeeksUTC()` instead of this method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>weeks</td>
<td>Number</td>
<td>The number of weeks to add. Use a negative number to subtract.</td>
</tr>
</tbody>
</table>
GlideDateTime - addWeeksLocalTime(Number weeks)

Adds a specified number of weeks to the current GlideDateTime object. A negative parameter subtracts weeks.

The method determines the local date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts weeks using the local date and time values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>weeks</td>
<td>Number</td>
<td>The number of weeks to add. Use a negative number to subtract.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `addWeeksLocalTime()` method in a scoped application, use the corresponding scoped method: `addWeeksLocalTime()`.

GlideDateTime - addWeeksUTC(Number weeks)

Adds a specified number of weeks to the current GlideDateTime object. A negative parameter subtracts weeks.
The method determines the UTC date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts weeks using the UTC date and time values.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>weeks</td>
<td>Number</td>
<td>The number of weeks to add. Use a negative number to subtract.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addWeeksUTC(-1);
gs.print(gdt.getDate());
```

Output: 2011-08-24

### Scoped equivalent

To use the addWeeksUTC() method in a scoped application, use the corresponding scoped method: addWeeksUTC().

### GlideDateTime - addMonths(Number months)

Adds a specified number of months to the current GlideDateTime object. A negative parameter subtracts months.

Use addMonthsLocalTime() or addMonthsUTC() instead of this method.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>months</td>
<td>Number</td>
<td>The number of months to add. Use a negative number to subtract.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
gdt.addMonths(2);
gs.print(gdt.getDate());

Output: 2011-10-31

**GlideDateTime - addMonthsLocalTime(Number months)**

Adds a specified number of months to the current GlideDateTime object. A negative parameter subtracts months.

The method determines the local date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts months using the local date and time values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>months</td>
<td>Number</td>
<td>The number of months to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addMonthsLocalTime(2);
gs.print(gdt.getDate());
```

Output: 2011-10-31

**Scoped equivalent**

To use the `addMonthsLocalTime()` method in a scoped application, use the corresponding scoped method: `addMonthsLocalTime()`.  

**GlideDateTime - addMonthsUTC(Number months)**

Adds a specified number of months to the current GlideDateTime object. A negative parameter subtracts months.

The method determines the UTC date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts months using the UTC date and time values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>months</td>
<td>Number</td>
<td>The number of months to add. Use a negative number to subtract.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addMonthsUTC(2);
gs.print(gdt.getDate());
```
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addMonthsUTC(2);
gs.print(gdt.getDate());
```

Output: 2011-10-31

### Scoped equivalent

To use the `addMonthsUTC()` method in a scoped application, use the corresponding scoped method: `addMonthsUTC()`.

### GlideDateTime - addYears(Number years)

Adds a specified number of years to the current GlideDateTime object. A negative parameter subtracts years.

Use `addYearsLocalTime()` or `addYearsUTC()` instead of this method.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>years</td>
<td>Number</td>
<td>The number of years to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2010-08-31 08:00:00");
gdt.addYears(1);
gs.print(gdt.getDate());
```

Output: 2011-08-31

### GlideDateTime - addYearsLocalTime(Number years)

Adds a specified number of years to the current GlideDateTime object. A negative parameter subtracts years.

The method determines the local date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts years using the local date and time values.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>years</td>
<td>Number</td>
<td>The number of years to add. To subtract use a negative value.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2010-08-31 08:00:00");
gdt.addYearsLocalTime(1);
gs.print(gdt.getDate());
```

Output: 2011-08-31

### Scoped equivalent

To use the `addYearsLocalTime()` method in a scoped application, use the corresponding scoped method: `AddYearsLocalTime()`.

### GlideDateTime - addYearsUTC(Number years)

Adds a specified number of years to the current GlideDateTime object. A negative parameter subtracts years.

The date and time value stored by GlideDateTime object is interpreted as being in the UTC time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>years</td>
<td>Number</td>
<td>The number of years to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2010-08-31 08:00:00");
gdt.addYearsUTC(1);
gs.print(gdt.getDate());
```
Scoped equivalent

To use the \texttt{addYearsUTC()} method in a scoped application, use the corresponding scoped method: \texttt{addYearsUTC()}.

\textbf{GlideDateTime - compareTo(Object dateTime)}

Compares two date and time objects to determine whether one occurs before the other or if they are equivalent.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
Name & Type & Description \\
\hline\hline
dateTime & Object & Date time in a GlideDateTime object \\
\hline
\end{tabular}
\end{table}

\textbf{Returns}

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
Type & Description \\
\hline\hline
Number & \begin{itemize}
\item 0 = Dates are equal
\item 1 = The object's date is after the date specified in the parameter
\item -1 = The object's date is before the date specified in the parameter
\end{itemize} \\
\hline
\end{tabular}
\end{table}

\begin{lstlisting}
var initDate = new GlideDateTime("2011-08-01 12:00:00");
var compDate1 = new GlideDateTime("2011-08-01 12:00:00");
var compDate2 = new GlideDateTime("2011-07-31 12:00:00");
var compDate3 = new GlideDateTime("2011-08-04 16:00:00");
gs.info(initDate.compareTo(compDate1)); // Equals 0
gs.info(initDate.compareTo(compDate2)); // initDate is after compDate2 (1)
gs.info(initDate.compareTo(compDate3)); // initDate is before compDate3 (-1)
\end{lstlisting}

Scoped equivalent

To use the \texttt{compareTo()} method in a scoped application, use the corresponding scoped method: \texttt{compareTo()}.

\textbf{GlideDateTime - equals(Object GDT)}

Compares an object with an existing value for equality.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDT</td>
<td>Object</td>
<td>The object to compare. Can be a GlideDateTime object or a valid date time string.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if they are equal, false otherwise.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 00:00:00");
gs.print(gdt.equals("2011-09-30 00:12:01"));
```

Output: false

### Scoped equivalent

To use the `equals()` method in a scoped application, use the corresponding scoped method: `equals()`.

### GlideDateTime - `getDate()`

Returns the date stored by the GlideDateTime object, expressed in the standard format, yyyy-MM-dd, and the system time zone, UTC by default.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDate</td>
<td>The date in the system time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getDate());
```

Output: 2011-08-31
Scoped equivalent

To use the `get_date()` method in a scoped application, use the corresponding scoped method: `get_date()`.

**GlideDateTime - getDayOfMonth()**

Returns the current day of the month in the UTC time zone.

Use `getDayOfMonthLocalTime()` and `getDayOfMonthUTC()` instead of this method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of the month in the UTC time zone, from 1 to 31.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-02 12:00:00");
gs.print(gdt.getDayOfMonth());
```

Output: 2

**GlideDateTime - getDayOfMonthLocalTime()**

Returns the day of the month stored by the GlideDateTime object, expressed in the current user's time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of the month in the user's time zone, from 1 to 31.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-02 12:00:00");
gs.print(gdt.getDayOfMonthLocalTime());
```
Output: 2

Scoped equivalent

To use the `getDayOfMonthLocalTime()` method in a scoped application, use the corresponding scoped method: `getDayOfMonthLocalTime()`.

GlideDateTime - `getDayOfMonthUTC()`

Returns the day of the month stored by the GlideDateTime object, expressed in the UTC time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of the month in the UTC time zone, from 1 to 31.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-02 12:00:00");
gs.print(gdt.getDayOfMonthUTC());
```

Output: 2

Scoped equivalent

To use the `getDayOfMonthUTC()` method in a scoped application, use the corresponding scoped method: `getDayOfMonthUTC()`.

GlideDateTime - `getDayOfWeek()`

Returns the day of the week stored by the GlideDateTime object, expressed in the user's time zone.

Use `getDayOfWeekLocalTime()` and `getDayOfWeekUTC()` instead of this method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of the week value - Monday = 1, ... Sunday = 7.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-01 12:00:00");
gs.print(gdt.getDayOfWeek());
```

Output: 4

**GlideDateTime - getDayOfWeekLocalTime()**

Returns the day of the week stored by the GlideDateTime object, expressed in the user's time zone.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of the week value - Monday = 1, ... Sunday = 7</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-01 12:00:00");
gs.print(gdt.getDayOfWeekLocalTime());
```

Output: 4

**Scoped equivalent**

To use the `getDayOfWeekLocalTime()` method in a scoped application, use the corresponding scoped method: `getDayOfWeekLocalTime()`.

**GlideDateTime - getDayOfWeekUTC()**

Returns the day of the week stored by the GlideDateTime object, expressed in the UTC time zone.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of the week value - Monday = 1, ... Sunday = 7</td>
</tr>
</tbody>
</table>

```js
var gdt = new GlideDateTime("2011-12-01 12:00:00");
gs.print(gdt.getDayOfWeekUTC());
```

Output: 4

### Scoped equivalent

To use the `getDayOfWeekUTC()` method in a scoped application, use the corresponding scoped method: `getDayOfWeekUTC()`.

**GlideDateTime - getDaysInMonth()**

Returns the number of days in the month stored by the GlideDateTime object, expressed in the Java Virtual Machine time zone.

Use `getDaysInMonthLocalTime()` and `getDaysInMonthUTC()` instead of this method.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of days in the current month in the Java Virtual Machine time zone.</td>
</tr>
</tbody>
</table>

```js
var gdt = new GlideDateTime(); //December
gs.print(gdt.getDaysInMonth());
```

Output: 31

**GlideDateTime - getDaysInMonthLocalTime()**

Returns the number of days in the month stored by the GlideDateTime object, expressed in the current user's time zone.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of days in the current month in the user's time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime(); //December
gs.print(gdt.getDaysInMonthLocalTime());
```

Output: 31

Scoped equivalent

To use the `getDaysInMonthLocalTime()` method in a scoped application, use the corresponding scoped method: `getDaysInMonthLocalTime()`.

**GlideDateTime - getDaysInMonthUTC()**

Returns the number of days in the month stored by the GlideDateTime object, expressed in the UTC time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of days in the month stored by the GlideDateTime object, expressed in the UTC time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime(); //December
gs.print(gdt.getDaysInMonthUTC());
```

Output: 31
Scoped equivalent

To use the `getDaysInMonthUTC()` method in a scoped application, use the corresponding scoped method: `getDaysInMonthUTC()`.

GlideDateTime - getDisplayValue()

Returns the date and time value in the current user's display format and time zone. Referring to the GlideDateTime object directly returns the date and time value in the GMT time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The date and time in the user's format and time zone. Keep in mind when designing business rules or script includes that this method may return values in different formats for different users.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getDisplayValue());
```

Output:

`2011-08-31 01:00:00`

Scoped equivalent

To use the `getDisplayValue()` method in a scoped application, use the corresponding scoped method: `getDisplayValue()`.

GlideDateTime - getDisplayValueInternal()

Returns the display value in the internal format (yyyy-MM-dd HH:mm:ss). This method is useful for date/time fields, but not for date fields.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The date and time values for the GlideDateTime object in the current user's time zone and the internal date and time format of yyyy-MM-dd HH:mm:ss.</td>
</tr>
</tbody>
</table>

```java
// Wednesday
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getDisplayValueInternal());
```

Output:

```
2011-08-31 01:00:00
```

Scoped equivalent

To use the `getDisplayValueInternal()` method in a scoped application, use the corresponding scoped method: `getDisplayValueInternal()`.

GlideDateTime - getDSTOffset()

Returns the amount of time that daylight saving time is offset.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Amount of time, in milliseconds, that daylight saving is offset. Returns 0 if there is no offset or if the time is not during daylight saving time.</td>
</tr>
</tbody>
</table>

```java
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getDSTOffset());
```

Output:

```
3600000
```
Scoped equivalent

To use the `getDSTOffset()` method in a scoped application, use the corresponding scoped method: `getDSTOffset()`.

**GlideDateTime - getErrorMsg()**

Returns the current error message.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The error message</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 aa:00:00"); // bad
gs.print(gdt.isValid()); // false
gs.print(gdt.getErrorMsg()); // reason
```

Output:

```
false
Could not parse DateTime: 2011-08-31 aa:00:00
```

Scoped equivalent

To use the `getErrorMsg()` method in a scoped application, use the corresponding scoped method: `getErrorMsg0`.

**GlideDateTime - getInternalFormattedLocalTime()**

Returns the object’s time in the local time zone and in the internal format.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The object’s time in the local time zone and the internal format.</td>
</tr>
</tbody>
</table>
**Scoped equivalent**

To use the `getInternalFormattedLocalTime()` method in a scoped application, use the corresponding scoped method: `getInternalFormattedLocalTime()`.

**GlideDateTime - getInternalMidnight(Number dayOfTheWeek)**

Returns a date and time object set to midnight of a specified day using UTC.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dayOfTheWeek</td>
<td>Number</td>
<td>The day of the week for which to return the date/time object.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDateTime</td>
<td>A GlideDateTime object set to midnight.</td>
</tr>
</tbody>
</table>

```java
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getInternalMidnight(2));
```

Output: 2011-08-30 00:00:01

**GlideDateTime - getLocalDate()**

Returns the date stored by the GlideDateTime object, expressed in the standard format, yyyy-MM-dd, and the current user’s time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDate</td>
<td>The date in the user's time zone.</td>
</tr>
</tbody>
</table>

```java
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getLocalDate());
```

Output: 2011-08-31
**Scoped equivalent**

To use the `getLocalDate()` method in a scoped application, use the corresponding scoped method: `getLocalDate()`.

**GlideDateTime - getLocalTime()**

Returns the time in the user's time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideTime</td>
<td>The time in the user's time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getLocalTime());
```

Output: 1970-01-01 01:00:00

**Scoped equivalent**

To use the `getLocalTime()` method in a scoped application, use the corresponding scoped method: `getLocalTime()`.

**GlideDateTime - getMonth()**

Returns the month stored by the GlideDateTime object, expressed in Java Virtual Machine time zone.

Use `getMonthLocalTime()` and `getMonthUTC()` instead of this method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The numerical value of the month, Jan=1, Dec=12.</td>
</tr>
</tbody>
</table>
var gdt = new GlideDateTime(); //December
gs.print(gdt.getMonth());

Output: 12

**GlideDateTime - getMonthLocalTime()**

Returns the month stored by the GlideDateTime object, expressed in the current user's time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The numerical value of the month, Jan=1, Dec=12.</td>
</tr>
</tbody>
</table>

var gdt = new GlideDateTime(); //December
gs.print(gdt.getMonthLocalTime());

Output: 12

**Scoped equivalent**

To use the `getMonthLocalTime()` method in a scoped application, use the corresponding scoped method: `getMonthLocalTime()`.

**GlideDateTime - getMonthUTC()**

Returns the month stored by the GlideDateTime object, expressed in the UTC time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The numerical value of the month, Jan=1, Dec=12.</td>
</tr>
</tbody>
</table>
```javascript
var gdt = new GlideDateTime(); //December
gs.print(gdt.getMonthUTC());
```

Output: 12

**Scoped equivalent**

To use the `getMonthUTC()` method in a scoped application, use the corresponding scoped method: `getMonthUTC()`.

**GlideDateTime - getNumericValue()**

Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of milliseconds since January 1, 1970, 00:00:00 GMT.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getNumericValue());
```

Output: 1314777600000

**GlideDateTime - getSpanTime(Number dayOfWeek)**

Returns the amount of time elapsed since the midnight of a specified day to the current time.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dayOfWeek</td>
<td>Number</td>
<td>Day of week value from 1 to 7, 1 = Monday, 7 = Sunday.</td>
</tr>
</tbody>
</table>
**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideTime</td>
<td>The amount of time elapsed since midnight of the specified day. To display the result in user-friendly terms, set the value to GlideDuration.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00"); //Wednesday
var dur = new GlideDuration();
var span = gdt.getSpanTime(1); //how much time since Monday 00:00:00
dur.setValue(span);
gs.print(dur.getDisplayValue());
```

Output: 2 Days 8 Hours

**GlideDateTime - getTime()**

Returns the Unix duration stamp.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideTime</td>
<td>The Unix duration stamp in system format based on GMT time.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getTime());
```

Output: 1970-01-01 08:00:00

**Scoped equivalent**

To use the `getTime()` method in a scoped application, use the corresponding scoped method: `getTime()`.

**GlideDateTime - getTZOffset()**

Returns the time zone offset in milliseconds.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of milliseconds of the time zone offset</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.getLocalTime(); // PST local time
gs.print(gdt.getTZOffset());
```

Output: -28800000

**Scoped equivalent**

To use the `getTZOffset()` method in a scoped application, use the corresponding scoped method: `getTZOffset()`.

**GlideDateTime - getUserFormattedLocalTime()**

Returns the object's time in local time zone in the user's format.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The object's time in local time and the user's format.</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `getUserFormattedLocalTime()` method in a scoped application, use the corresponding scoped method: `getUserFormattedLocalTime()`.

**GlideDateTime - getUserTimeZone()**

Returns the time zone for the current user session.

This method is equivalent to `gs.getSession().getTimeZone()`.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeZone</td>
<td>TimeZone object for the current user.</td>
</tr>
</tbody>
</table>

```
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getUserTimeZone());
```

Output:

```
sun.util.calendar.ZoneInfo[id="America/Los_Angeles",offset=-28800000,dstSavings=3600000,useDaylight=true,transitions=185,lastRule=java.util.SimpleTimeZone[id=America/Los_Angeles,offset=-28800000,dstSavings=3600000,useDaylight=true,startYear=0,startMonth=2,startDay=8,startDayOfWeek=1,startTime=7200000,startTimeMode=0,endMonth=10,endDay=1,endDayOfWeek=1,endTime=7200000,endTimeMode=0]]
```

**GlideDateTime - getUTCMidnight(Number dayOfTheWeek)**

Returns a GlideDateTime object with the time set to midnight using the UTC time zone.

This method sets the date of the new GlideDateTime object as the specified day of the week within the week of the original GlideDateTime object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dayOfTheWeek</td>
<td>Number</td>
<td>The day of the week, from 1 to 7. Monday=1, Sunday=7. Do not enter 0 in this parameter.</td>
</tr>
</tbody>
</table>

```
var gdt = new GlideDateTime("2011-08-31 08:00:00");
Wednesday, 3rd day of the week.
gs.print(gdt.getUTCMidnight(5)); //Friday, 5th day of the week.
```

Output: 2011-09-02 00:00:00
**GlideDateTime - getValue()**

Returns the date and time value stored by the GlideDateTime object in the internal format, `yyyy-MM-dd HH:mm:ss`, and the system time zone, UTC by default.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The date and time in the internal format and system time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.getValue());
```

Output:

```
2011-08-31 08:00:00
```

**Scoped equivalent**

To use the `getValue()` method in a scoped application, use the corresponding scoped method: `getValue()`.

**GlideDateTime - getWeekOfYearLocalTime()**

Returns the number of the week stored by the GlideDateTime object, expressed in the current user's time zone.

All weeks begin on Sunday. The first week of the year is the week that contains at least one day of the new year. The week beginning Sunday 2015-12-27 is considered the first week of 2016 as that week contains January 1 and 2.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of the current week. The highest week number in a year is either 52 or 53.</td>
</tr>
</tbody>
</table>
var gdt = new GlideDateTime("2011-12-01 12:00:00");
gs.print(gdt.getWeekOfYearUTC());

Output: 49

 Scoped equivalent

To use the `getWeekOfYearLocalTime()` method in a scoped application, use the corresponding scoped method: `getWeekOfYearLocalTime()`.

**GlideDateTime - getWeekOfYearUTC()**

Returns the number of the week stored by the GlideDateTime object, expressed in the UTC time zone.

All weeks begin on Sunday. The first week of the year is the week that contains at least one day of the new year. The week beginning Sunday 2015-12-27 is considered the first week of 2016 as that week contains January 1 and 2.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of the current week in UTC time. The highest week number in a year is either 52 or 53.</td>
</tr>
</tbody>
</table>

var gdt = new GlideDateTime("2011-12-01 12:00:00");
gs.print(gdt.getWeekOfYearUTC());

Output: 49

 Scoped equivalent

To use the `getWeekOfYearUTC()` method in a scoped application, use the corresponding scoped method: `getWeekOfYearUTC()`.

**GlideDateTime - getYear()**

Returns the year stored by the GlideDateTime object, expressed in the Java Virtual Machine time zone.

Use `getYearLocalTime()` and `getYearUTC()` instead of this method.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The 4-digit year value in the Java Virtual Machine time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime(); //2011
gs.print(gdt.getYear());
```

Output: 2011

**GlideDateTime - getYearLocalTime()**

Returns the year stored by the GlideDateTime object, expressed in the current user's time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The 4-digit year value in the user's time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime(); //2011
gs.print(gdt.getYearLocalTime());
```

Output: 2011

**Scoped equivalent**

To use the `getYearLocalTime()` method in a scoped application, use the corresponding scoped method: `getYearLocalTime()`.

**GlideDateTime - getYearUTC()**

Returns the year stored by the GlideDateTime object, expressed in the UTC time zone.
### GlideDateTime

#### GlideDateTime()

Instantiates a new GlideDateTime object with the current date and time in GMT format.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
var gdt = new GlideDateTime(); //2011
gs.print(gdt.getYearUTC());
```

Output: 2011

#### GlideDateTime - GlideDateTime(String dateTime)

Instantiates a new GlideDateTime object from a date and time value in the UTC time zone specified with the format yyyy-MM-dd HH:mm:ss.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateTime</td>
<td>String</td>
<td>A UTC date and time using the format yyyy-MM-dd HH:mm:ss.</td>
</tr>
</tbody>
</table>

```
var gdt = new GlideDateTime();
```
```javascript
var gdt = new GlideDateTime("2011-01-01 12:00:00");
```

**GlideDateTime - GlideDateTime(GlideDateTime gDT)**

Instantiates a new GlideDateTime object set to the time of a specified GlideDateTime object in GMT format.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gDT</td>
<td>GlideDateTime</td>
<td>Object used to set the time of the new object.</td>
</tr>
</tbody>
</table>

```javascript
var start = new GlideDateTime("2011-01-01 12:00:00");
var end = new GlideDateTime(start);
gs.print(end);
```

Output:

```
2011-01-01 12:00:00
```

**GlideDateTime - hasDate()**

Determines if an object's date is set.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the object's date is set, false otherwise.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.hasDate());
```

Output:

```
true
```
Scoped equivalent

To use the `hasDate()` method in a scoped application, use the corresponding scoped method: `hasDate()`.

**GlideDateTime - isDST()**

Determines if the object's time uses a daylight saving offset

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the time is daylight saving time, false otherwise.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 00:00:00");
gs.print(gdt.isDST()); //true
```

Output:

true

Scoped equivalent

To use the `isDST()` method in a scoped application, use the corresponding scoped method: `isDST()`.

**GlideDateTime - isValid()**

Determines if a value is a valid date and time.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if value is valid, false otherwise.</td>
</tr>
</tbody>
</table>
var gdt = new GlideDateTime("2011-08-31 00:00:00"); //bad
gs.print(gdt.isValid()); //false

Output: false

Scoped equivalent

To use the `isValid()` method in a scoped application, use the corresponding scoped method: `isValid()`.

GlideDateTime - `setDayOfMonth(Number day)`

Sets the day of the month to a specified value.

Use `setDayOfMonthLocalTime(day)` and `setDayOfMonthUTC(day)` instead of this method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>day</td>
<td>Number</td>
<td>Day of the month, from 1 to 31.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var gdt = new GlideDateTime();
gdt.setDayOfMonth(9);
gs.print(gdt.getDayOfMonth());

Output: 9

GlideDateTime - `setDayOfMonthLocalTime(Number day)`

Sets the day of the month to a specified value in the current user's time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>day</td>
<td>Number</td>
<td>The day of month to change to, from 1 to 31. If this value is greater than the maximum number of days in the month, the value is set to the last day of the month.</td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.setDayOfMonthLocalTime(9);
gs.print(gdt.getDayOfMonthLocalTime());
```

Output: 9

### Scoped equivalent

To use the `setDayOfMonthLocalTime()` method in a scoped application, use the corresponding scoped method: `setDayOfMonthLocalTime()`.

**GlideDateTime - setDayOfMonthUTC(Number day)**

Sets the day of the month to a specified value in the UTC time zone.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>day</td>
<td>Number</td>
<td>The day of month to change to, from 1 to 31. If this value is greater than the maximum number of days in the month, the value is set to the last day of the month.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.setDayOfMonthUTC(9);
gs.print(gdt.getDayOfMonthUTC());
```

Output: 9

### Scoped equivalent

To use the `setDayOfMonthUTC()` method in a scoped application, use the corresponding scoped method: `setDayOfMonthUTC()`.
GlideDateTime - setDisplayValue(String asDisplayed)

Sets a date and time value using the current user's display format and time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>asDisplayed</td>
<td>String</td>
<td>The date and time in the current user's display format and time zone. The parameter must be formatted using the current user's preferred display format, such as MM-dd-yyyy HH:mm:ss. To assign the current date and time to a variable in a workflow script, use &lt;variable&gt;.setDisplayValue(gs.nowDateTime);.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-02-02 12:00:00");
gdt.setDisplayValue("2011-01-01 12:00:00");
gs.print(gdt.getValue());
```

Output:

2011-01-01 20:00:00

Scoped equivalent

To use the setDisplayValue() method in a scoped application, use the corresponding scoped method: setDisplayValue().

GlideDateTime - setDisplayValue(String dateTime, String format)

Sets a date and time value using the current user's time zone and the specified date and time format.

This method throws a runtime exception if the date and time format used in the dateTime parameter does not match the format parameter. You can retrieve the error message by calling getErrorMsg() on the GlideDateTime object after the exception is caught.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateTime</td>
<td>String</td>
<td>The date and time in the current user's time zone.</td>
</tr>
<tr>
<td>format</td>
<td>String</td>
<td>The format to use to parse the dateTime parameter.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-02-02 12:00:00");
gdt.setDisplayValue("20-5-2011 12:00:00", "dd-MM-yyyy HH:mm:ss");
gs.print(gdt.getValue());
```

Output:

```
2011-05-20 19:00:00
```

Scoped equivalent

To use the `setDisplayValue()` method in a scoped application, use the corresponding scoped method: `setDisplayValue()`.

GlideDateTime - `setDisplayValueInternal(String dateTime)`

Sets a date and time value using the internal format (yyyy-MM-dd HH:mm:ss) and the current user's time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateTime</td>
<td>String</td>
<td>The date and time in internal format</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-02-02 12:00:00");
```
 Scoped equivalent

To use the `setDisplayValueInternal()` method in a scoped application, use the corresponding scoped method: `setDisplayValueInternal()`.  

**GlideDateTime - setDisplayValueInternalWithAlternates(String dateTime)**  
Sets a date and time value using the internal format (yyyy-MM-dd HH:mm:ss) and the current user's time zone.  
This method attempts to parse incomplete date and time values.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>dateTime</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>void</td>
</tr>
</tbody>
</table>

**GlideDateTime - setGlideDateTime(GlideDateTime gDT)**  
Sets the date and time of the current object using an existing GlideDateTime object.  
This method is equivalent to instantiating a new object with a GlideDateTime parameter.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>gDT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>void</td>
</tr>
</tbody>
</table>
GlideDateTime - setGlideDateTime(String dateTime)

This method is equivalent to setValue(Object).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateTime</td>
<td>String</td>
<td>The date and time to use. Accepts either a string in the GMT time zone in the internal format, or a GlideDateTime object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideDateTime - setInitialValue(String dateTime)

Sets the date and time.

This method is equivalent to setValue(Object).

```
var gdt = new GlideDateTime();
gdt.setInitialValue("2011-01-01 12:00:00");
gs.print(gdt.getValue());
```

Output: 2011-01-01 12:00:00

GlideDateTime - setMonth(Number month)

Sets the month stored by the GlideDateTime object to a specified value using the Java Virtual Machine time zone.

Use setMonthLocalTime() or setMonthUTC() instead of this method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>month</td>
<td>Number</td>
<td>The month to change to.</td>
</tr>
</tbody>
</table>
GlideDateTime - setMonthLocalTime(Number month)
Sets the month stored by the GlideDateTime object to a specified value using the current user's time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>month</td>
<td>Number</td>
<td>The month to change to.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideDateTime - setMonthUTC(Number month)
Sets the month stored by the GlideDateTime object to a specified value using the UTC time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>month</td>
<td>Number</td>
<td>The month to change to.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the setMonthLocalTime() method in a scoped application, use the corresponding scoped method: setMonthLocalTime().
var gdt = new GlideDateTime();
gdt.setMonthUTC(1);
gs.print(gdt.getMonthUTC());

Output: 1

**Scoped equivalent**

To use the `setMonthUTC()` method in a scoped application, use the corresponding scoped method: `setMonthUTC()`.

**GlideDateTime - setNumericValue(Number milliseconds)**

Sets the date and time to the number of milliseconds since January 1, 1970 00:00:00 GMT.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>Number</td>
<td>Number of milliseconds</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var gdt = new GlideDateTime();
gdt.setNumericValue(1314777600000);
gs.print(gdt.getValue());

Output: 2011-08-31 08:00:00

**GlideDateTime - setTZ(TimeZone timeZone)**

Sets the time zone of the GlideDateTime object to be the specified time zone.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeZone</td>
<td>TimeZone</td>
<td>A time zone object</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var tz = gs.getSession().getTimeZone();
var gdt = new GlideDateTime();
gdt.setTZ(tz);
```

**GlideDateTime - setValue(Object dateTime)**

Sets the date and time of the GlideDateTime object.
This method is equivalent to `setInitialValue()`. 
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateTime</td>
<td>Object</td>
<td>The date and time to use. This parameter may be one of several types.</td>
</tr>
</tbody>
</table>

- A string in the UTC time zone and the internal format of `yyyy-MM-dd HH:mm:ss`: Sets the value of the object to the specified date and time. Using the method this way is equivalent to instantiating a new GlideDateTime object using the `GlideDateTime(value)` constructor. If the date and time format used does not match the internal format, the method attempts to set the date and time using other available formats. Resolving the date and time this way can lead to inaccurate data due to ambiguity in the day and month values. When using a non-standard date and time format, use `setValue(dt, format)` instead.
- A GlideDateTime object: Sets the value of the object to the date and time stored by the GlideDateTime passed in the parameter. Using the method this way is equivalent to instantiating a new GlideDateTime object using the `GlideDateTime(g)` constructor.
- A Java Date object: Sets the value of the object using the value stored by the Java Date object. Using the method this way is equivalent to passing the value returned by the Java Date object `getTime()` to the GlideDateTime `setNumericValue()` method. This method does not accept JavaScript Date objects.
- A JavaScript Number: Sets the value of the object using the Number value as milliseconds past January 1, 1970 00:00:00 GMT. Using the method this way is equivalent to the `setNumericValue(milliseconds)` method.
- A Java Integer or Long: Sets the value of the object using the Integer or Long value as milliseconds past January 1, 1970 00:00:00 GMT. Using the method this way is equivalent to the `setNumericValue(milliseconds)` method.
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-01-01 12:00:00");
gdt.setValue("2011-02-02 08:00:00");
gs.print(gdt.getValue());
```

Output:

`2011-02-02 08:00:00`

Scoped equivalent

To use the `setValue()` method in a scoped application, use the corresponding scoped method: `setValueUTC()`.

**GlideDateTime - setValueUTC(String dateTime, String format)**

Sets a date and time value using the UTC time zone and the specified date and time format.

This method throws a runtime exception if the date and time format used in the `dateTime` parameter does not match the format parameter. You can retrieve the error message by calling `getErrorMsg()` on the GlideDateTime object after the exception is caught.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateTime</td>
<td>String</td>
<td>The date and time to use.</td>
</tr>
<tr>
<td>format</td>
<td>String</td>
<td>The format to use.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-01-01 12:00:00");
gdt.setValueUTC("15-02-2011 08:00:00", "dd-MM-yyyy HH:mm:ss");
gs.print(gdt.getValue());
```

Output:

`2011-02-15 08:00:00`
Scoped equivalent

To use the `setValueUTC()` method in a scoped application, use the corresponding scoped method: `setValueUTC()`.

**GlideDateTime - setYear(Number year)**

Sets the year stored by the GlideDateTime object to a specified value using the Java Virtual Machine time zone.

Use `setYearLocalTime()` or `setYearUTC()` instead of this method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>Number</td>
<td>The year to change to.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
var gdt = new GlideDateTime();
gdt.setYear(2013);
gs.print(gdt.getYear());
```

Output: 2013

**GlideDateTime - setYearLocalTime(Number year)**

Sets the year stored by the GlideDateTime object to a specified value using the current user's time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>Number</td>
<td>The year to change to.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
var gdt = new GlideDateTime();
gdt.setYearLocalTime(2013);
gs.print(gdt.getYearLocalTime());
```
Output: 2013

Scoped equivalent

To use the `setYearLocalTime()` method in a scoped application, use the corresponding scoped method: `setYearLocalTime()`.

**GlideDateTime - setYearUTC(Number year)**

Sets the year stored by the GlideDateTime object to a specified value using the UTC time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>Number</td>
<td>The year to change to.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.setYearUTC(2013);
gs.print(gdt.getYearUTC());
```

Output: 2013

Scoped equivalent

To use the `setYearUTC()` method in a scoped application, use the corresponding scoped method: `setYearUTC()`.

**GlideDateTime - subtract(GlideTime time)**

Subtracts a specified amount of time.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>GlideTime</td>
<td>The time to subtract</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
```
```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
var gtime1 = new GlideTime();
gtime1.setValue("00:00:20");
gdt.subtract(gtime1);
gs.print(gdt.getTime());
```

Output: 1970-01-01 07:59:40

**Scoped equivalent**

To use the `subtract()` method in a scoped application, use the corresponding scoped method: `subtract()`.

### GlideDateTime - subtract(Number milliseconds)

Subtracts a specified number of milliseconds from the GlideDateTime object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>Number</td>
<td>The number of milliseconds to subtract</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-07 08:00:00");
gdt.subtract(1000);
gs.print(gdt.getValue());
```

Output: 2011-12-07 07:59:59

**Scoped equivalent**

To use the `subtract()` method in a scoped application, use the corresponding scoped method: `subtract()`.

### GlideDateTime - subtract(GlideDateTime start, GlideDateTime end)

Returns the duration difference between two GlideDateTime values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>GlideDateTime</td>
<td>The start value</td>
</tr>
</tbody>
</table>
### GlideDateTime

**GlideDateTime - toString()**

Returns the date and time value stored by the GlideDateTime object in the internal format, `yyyy-MM-dd HH:mm:ss`, and the system time zone, UTC by default.

This method is equivalent to `getValue()`.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The date and time stored by the GlideDateTime object in the system time zone and format.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.toString());
```

Output: 2011-08-31 08:00:00

---

**Scoped equivalent**

To use the `toString()` method in a scoped application, use the corresponding scoped method: `toString()`.

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.toString());
```

Output: 2011-08-31 08:00:00
GlideDateTime

The scoped GlideDateTime class provides methods for performing operations on GlideDateTime objects, such as instantiating GlideDateTime objects or working with glide_date_time fields.

Use the GlideDateTime methods to perform date-time operations, such as instantiating a GlideDateTime object, performing date-time calculations, formatting a date-time, or converting between date-time formats.

Scoped GlideDateTime - add(GlideTime gd)

Adds a GlideTime object to the current GlideDateTime object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gd</td>
<td>GlideTime</td>
<td>The GlideTime object to add.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
var gtime1 = new GlideTime();
gtime1.setValue("00:00:20");
gdt.add(gtime1);
var gtime2 = gdt.getTime();
gs.info(gtime2.getByFormat('hh:mm:ss'));
```

Scoped GlideDateTime - add(Number milliseconds)

Adds the specified number of milliseconds to the current GlideDateTime object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>Number</td>
<td>The number of milliseconds to add.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.info(gdt.getNumericValue());
gdt.add(10);
```
Scoped GlideDateTime - addDaysLocalTime(Number days)

Adds a specified number of days to the current GlideDateTime object. A negative parameter subtracts days. The method determines the local date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts days using the local date and time values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>Number</td>
<td>The number of days to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addDaysLocalTime(-1);
gs.info(gdt.getLocalDate());
```

Output: 2011-08-30

Scoped GlideDateTime - addDaysUTC(Number days)

Adds a specified number of days to the current GlideDateTime object. A negative parameter subtracts days. The method determines the UTC date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts days using the UTC date and time values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>Number</td>
<td>The number of days to add. Use a negative number to subtract.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addDaysUTC(-1);
gs.info(gdt.getDate());
```
Output: 2011-08-30

Scoped GlideDateTime - addMonthsLocalTime(Number months)

Adds a specified number of months to the current GlideDateTime object. A negative parameter subtracts months. The method determines the local date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts months using the local date and time values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>months</td>
<td>Number</td>
<td>The number of months to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addMonthsLocalTime(2);
gs.info(gdt.getDate());
```

Output: 2011-10-31

Scoped GlideDateTime - addMonthsUTC(Number months)

Adds a specified number of months to the current GlideDateTime object. A negative parameter subtracts months. The method determines the UTC date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts months using the UTC date and time values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>months</td>
<td>Number</td>
<td>The number of months to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addMonthsUTC(2);
```
 Scoped GlideDateTime - addSeconds(Number seconds)

Adds the specified number of seconds to the current GlideDateTime object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>seconds</td>
<td>Number</td>
<td>The number of seconds to add.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-07 08:00:00");
gdt.addSeconds(1000);
gs.info(gdt.getValue());
```

Output: 2011-12-07 08:16:40

 Scoped GlideDateTime - addWeeksLocalTime(Number weeks)

Adds a specified number of weeks to the current GlideDateTime object. A negative parameter subtracts weeks. The method determines the local date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts weeks using the local date and time values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>weeks</td>
<td>Number</td>
<td>The number of weeks to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addWeeksLocalTime(-1);
gs.info(gdt.getDate());
```

Output: 2011-08-24
Scoped GlideDateTime - addWeeksUTC(Number weeks)

Adds a specified number of weeks to the current GlideDateTime object. A negative parameter subtracts weeks. The method determines the UTC date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts weeks using the UTC date and time values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>weeks</td>
<td>Number</td>
<td>The number of weeks to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gdt.addWeeksUTC(-1);
gs.info(gdt.getDate());
```

Output: 2011-08-24

Scoped GlideDateTime - addYearsLocalTime(Number years)

Adds a specified number of years to the current GlideDateTime object. A negative parameter subtracts years. The method determines the local date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts years using the local date and time values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>years</td>
<td>Number</td>
<td>The number of years to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2010-08-31 08:00:00");
gdt.addYearsLocalTime(1);
gs.info(gdt.getDate());
```

Output: 2011-08-31
Scoped GlideDateTime - addYearsUTC(Number years)

Adds a specified number of years to the current GlideDateTime object. A negative parameter subtracts years. The date and time value stored by GlideDateTime object is interpreted as being in the UTC time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>years</td>
<td>Number</td>
<td>The number of years to add. Use a negative value to subtract.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2010-08-31 08:00:00");
gdt.addYearsUTC(1);
gs.info(gdt.getDate());
```

Output: 2011-08-31

Scoped GlideDateTime - after(GlideDateTime gdt)

Determines if the GlideDateTime object occurs after the specified GlideDateTime.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gdt</td>
<td>GlideDateTime</td>
<td>The time to check against.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the GlideDateTime object's time is after the time specified by the parameter.</td>
</tr>
</tbody>
</table>

```javascript
var gdt1 = new GlideDateTime("2016-05-09 10:11:12");
var gdt2 = new GlideDateTime("2017-06-12 15:11:12");
gs.info(gdt1.after(gdt2));
```

Output: False

 Scoped GlideDateTime - before(GlideDateTime gdt)

Determines if the GlideDateTime object occurs before the specified GlideDateTime.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gdt</td>
<td>GlideDateTime</td>
<td>The time to check against.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the GlideDateTime object’s time is before the time specified by the parameter.</td>
</tr>
</tbody>
</table>

```javascript
var gdt1 = new GlideDateTime("2016-05-09 10:11:12");
var gdt2 = new GlideDateTime("2017-06-12 15:11:12");
gs.info(gdt1.before(gdt2));
```

Output: True

### Scoped GlideDateTime - compareTo(Object o)

Compares two date and time objects to determine whether they are equivalent or one occurs before or after the other.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Object</td>
<td>Date and time object in GlideDateTime format</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Number     | • 0 = Dates are equal  
• 1 = The object’s date is after the date specified in the parameter  
• -1 = The object’s date is before the date specified in the parameter |

```javascript
var initDate = new GlideDateTime("2011-08-01 12:00:00");
var compDate1 = new GlideDateTime("2011-08-01 12:00:00");
var compDate2 = new GlideDateTime("2011-07-31 12:00:00");
var compDate3 = new GlideDateTime("2011-08-04 16:00:00");
gs.info(initDate.compareTo(compDate1)); // Equals (0)
gs.info(initDate.compareTo(compDate2)); // initDate is after compDate2 (1)
```
Scoped GlideDateTime - equals(Object dateTime)
Compared a datetime with an existing value for equality.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateTime</td>
<td>GlideDateTime object or String</td>
<td>The datetime to compare.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if they are equal; otherwise, false.</td>
</tr>
</tbody>
</table>

```java
var gdt = new GlideDateTime("2011-08-31 00:00:00");
gs.info(gdt.equals("2011-09-30 00:12:01"));
```

Output: false

Scoped GlideDateTime - getDate()
Gets the date stored by the GlideDateTime object, expressed in the standard format, yyyy-MM-dd, and the system time zone, UTC by default.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDate</td>
<td>The date in the system time zone.</td>
</tr>
</tbody>
</table>

```java
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.info(gdt.getDate());
```

Scoped GlideDateTime - getDayOfMonthUTC()
Gets the day of the month stored by the GlideDateTime object, expressed in the UTC time zone.

```java
var gdt = new GlideDateTime("2011-08-31 00:00:00");
gs.info(gdt.getDayOfMonthUTC());
```
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of the month in the UTC time zone, from 1 to 31.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-02 12:00:00");
gs.info(gdt.getDayOfMonthUTC());
```

Output: 02

**Scoped GlideDateTime - getDayOfMonthLocalTime()**

Gets the day of the month stored by the GlideDateTime object, expressed in the current user's time zone.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of the month in the user's time zone, from 1 to 31.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-02 12:00:00");
gs.info(gdt.getDayOfMonthLocalTime());
```

### Scoped GlideDateTime - getDayOfWeekUTC()

Gets the day of the week stored by the GlideDateTime object, expressed in the UTC time zone.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of the week in the UTC time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-02 12:00:00");
gs.info(gdt.getDayOfWeekUTC());
```
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of week value from 1 to 7. Monday equals 1, Sunday equals 7.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-01 12:00:00"); // Thursday
gs.info(gdt.getDayOfWeekLocalTime());
```

**Scoped GlideDateTime - getDayOfWeekLocalTime()**

Gets the day of the week stored by the GlideDateTime object, expressed in the user's time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day of week value, in the user's time zone, from 1 to 7. Monday equals 1, Sunday equals 7.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-01 12:00:00"); // Thursday
gs.info(gdt.getDayOfWeekLocalTime());
```

**Scoped GlideDateTime - getDaysInMonthLocalTime()**

Gets the number of days in the month stored by the GlideDateTime object, expressed in the current user's time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of days in the current month in the user's time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-02 12:00:00"); //December
gs.info(gdt.getDaysInMonthLocalTime());
```

**Scoped GlideDateTime - getDaysInMonthUTC()**

Gets the number of days in the month stored by the GlideDateTime object, expressed in the UTC time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of days in the month stored by the GlideDateTime object, expressed in the UTC time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-11-02 12:00:00"); //November
gs.info(gdt.getDaysInMonthUTC());
```

**Scoped GlideDateTime - getDisplayValue()**

Gets the date and time value in the current user's display format and time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The date and time in the user's format and time zone. Keep in mind when designing business rules or script includes that this method may return values in different formats for different users.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.info(gdt.getDisplayValue()); //uses current user session time zone (US/Pacific)
```

**Scoped GlideDateTime - getDisplayValueInternal()**

Gets the display value in the internal format (yyyy-MM-dd HH:mm:ss).

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The date and time values for the GlideDateTime object in the current user's time zone and the internal date and time format of yyyy-MM-dd HH:mm:ss.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.info(gdt.getDisplayValueInternal()); //uses current user session time zone (US/Pacific)
```

**Scoped GlideDateTime - getDSTOffset()**

Gets the amount of time that daylight saving time is offset.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Amount of time, in milliseconds, that daylight saving is offset. Returns 0 if there is no offset or if the time is not during daylight saving time.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2014-08-31 08:00:00");
gs.info(gdt.getDSTOffset()); // uses current user session time zone (US/Pacific)
```

**Scoped GlideDateTime - getErrorMsg()**

Gets the current error message.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The error message.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.setDisplayValue("2011-aa-01 00:00:00");
gs.info(gdt.getErrorMsg());
```

Output: Could not parse DateTime: 2011-aa-01 00:00:00

**Scoped GlideDateTime - getInternalFormattedLocalTime()**

Returns the object's time in the local time zone and in the internal format.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The object’s time in the local time zone and the internal format.</td>
</tr>
</tbody>
</table>
Scoped GlideDateTime - getLocalDate()

Gets the date stored by the GlideDateTime object, expressed in the standard format, yyyy-MM-dd, and the current user's time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDate</td>
<td>The date in the user's time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.info(gdt.getLocalDate());
```

Scoped GlideDateTime - getLocalTime()

Returns a GlideTime object that represents the time portion of the GlideDateTime object in the user's time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideTime</td>
<td>The time in the user's time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2014-08-31 08:00:00");
gt = gdt.getLocalTime();
gs.info("local time is " + gt.getByFormat('hh:mm:ss'));
```

Scoped GlideDateTime - getMonthLocalTime()

Gets the month stored by the GlideDateTime object, expressed in the current user's time zone.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The numerical value of the month.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-11-02 12:00:00"); //November
gs.info(gdt.getMonthLocalTime());
```

Scoped GlideDateTime - getMonthUTC()

Gets the month stored by the GlideDateTime object, expressed in the UTC time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The numerical value of the month.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-11-02 12:00:00"); //November
gs.info(gdt.getMonthUTC());
```

Scoped GlideDateTime - getNumericValue()

Gets the number of milliseconds since January 1, 1970, 00:00:00 GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of milliseconds since January 1, 1970, 00:00:00 GMT.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.info(gdt.getNumericValue());
```

**Scoped GlideDateTime - getTime()**

Returns a GlideTime object that represents the time portion of the GlideDateTime object.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideTime</td>
<td>The Unix duration stamp in system format based on GMT time.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2014-08-31 08:00:00");
gt = gdt.getTime();
gs.info(gt.getByFormat('hh:mm:ss'));
```

**Scoped GlideDateTime - getTZOffset()**

Gets the time zone offset in milliseconds.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of milliseconds of time zone offset.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
```
gdt.getLocalTime(); // PST local time
gs.info(gdt.getTZoffset());

Output: -25200000

**Scoped GlideDateTime - getUserFormattedLocalTime()**

Returns the object's time in the local time zone and in the user's format.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The object's time in the local time zone and in the user's format.</td>
</tr>
</tbody>
</table>

**Scoped GlideDateTime - getValue()**

Gets the date and time value stored by the GlideDateTime object in the internal format, yyyy-MM-dd HH:mm:ss, and the system time zone, UTC by default.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The date and time value in the internal format and system time zone.</td>
</tr>
</tbody>
</table>

var gdt = new GlideDateTime("2014-08-31 08:00:00");
gs.info(gdt.getValue());

**Scoped GlideDateTime - getWeekOfYearUTC()**

Gets the number of the week stored by the GlideDateTime object, expressed in the UTC time zone. All weeks begin on Sunday. The first week of the year is the week that contains at least one day of the new year. The week beginning Sunday 2015-12-27 is considered the first week of 2016 as that week contains January 1 and 2.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number  - The number of the current week in UTC time. The highest week number in a year is either 52 or 53.</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-01 12:00:00"); // 49th week, 1st week in december
gs.info(gdt.getWeekOfYearUTC());
```

Scoped GlideDateTime - getWeekOfYearLocalTime()

Gets the number of the week stored by the GlideDateTime object, expressed in the current user's time zone. All weeks begin on Sunday. The first week of the year is the week that contains at least one day of the new year. The week beginning Sunday 2015-12-27 is considered the first week of 2016 as that week contains January 1 and 2.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number  - The number of the current week in local time. The highest week number in a year is either 52 or 53.</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-01 12:00:00"); // 49th week, 1st week in december
gs.info(gdt.getWeekOfYearLocalTime());
```

Scoped GlideDateTime - getYearLocalTime()

Gets the year stored by the GlideDateTime object, expressed in the current user's time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Four-digit year value in the user's time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-11-02 12:00:00");
gs.info(gdt.getYearLocalTime());
```

### Scoped GlideDateTime - getYearUTC()

Gets the year stored by the GlideDateTime object, expressed in the UTC time zone.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>4-digit year value in the UTC time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-11-02 12:00:00");
gs.info(gdt.getYearUTC());
```

### Scoped GlideDateTime - GlideDateTime()

Instantiates a new GlideDateTime object with the current date and time in Greenwich Mean Time (GMT).

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
```

### Scoped GlideDateTime - GlideDateTime(GlideDateTime g)

Instantiates a new GlideDateTime object set to the time of the GlideDateTime object passed in the parameter.

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>GlideDateTime</td>
<td>The GlideDateTime object to use for setting the time of the new object.</td>
</tr>
</tbody>
</table>

```javascript
var start = new GlideDateTime("2011-01-01 12:00:00");
var end = new GlideDateTime(start);
gs.info(end);
```

**Scoped GlideDateTime - GlideDateTime(String value)**

Instantiates a new GlideDateTime object from a date and time value in the UTC time zone specified with the format yyyy-MM-dd HH:mm:ss.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>String</td>
<td>A UTC date and time using the internal format yyyy-MM-dd HH:mm:ss.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-01-01 12:00:00");
```

**Scoped GlideDateTime - hasDate()**

Determines if an object's date is set.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the object date is set; otherwise, returns false.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.info(gdt.hasDate());
```
Scoped GlideDateTime - isDST()
Determines if an object's time uses a daylight saving offset.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the time is daylight saving; otherwise, returns false.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2014-08-31 00:00:00");
gs.info(gdt.isDST()); //true
```

Scoped GlideDateTime - isValid()
Determines if a value is a valid date and time.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if value is valid; otherwise, returns false.</td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.setDisplayValue("2011-aa-01 00:00:00");
gs.info(gdt.isValid());
```

Output: false

Scoped GlideDateTime - onOrAfter(GlideDateTime gdt)
Determines if the GlideDateTime object occurs on or after the specified GlideDateTime.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gdt</td>
<td>GlideDateTime</td>
<td>The time to check against.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the GlideDateTime object's time is on or after the time specified by the parameter.</td>
</tr>
</tbody>
</table>

```javascript
var gdt1 = new GlideDateTime("2016-05-09 10:11:12");
var gdt2 = new GlideDateTime("2017-06-12 15:11:12");
gs.info(gdt1.onOrAfter(gdt2));
```

Output: False

**Scoped GlideDateTime - onOrBefore(GlideDateTime gdt)**

Determines if the GlideDateTime object occurs on or before the specified GlideDateTime.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gdt</td>
<td>GlideDateTime</td>
<td>The time to check against.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the GlideDateTime object's time is on or before the time specified by the parameter.</td>
</tr>
</tbody>
</table>

```javascript
var gdt1 = new GlideDateTime("2016-05-09 10:11:12");
var gdt2 = new GlideDateTime("2017-06-12 15:11:12");
gs.info(gdt1.onOrBefore(gdt2));
```

Output: True

**Scoped GlideDateTime - setDayOfMonthLocalTime(Number day)**

Sets the day of the month to a specified value in the current user's time zone.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>day</td>
<td>Number</td>
<td>The day of month to change to, from 1 to 31. If this value is greater than</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the maximum number of days in the month, the value is set to the last day of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the month.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.setDayOfMonthLocalTime(9);
gs.info(gdt.getDayOfMonthLocalTime());
```

Scoped GlideDateTime - setDayOfMonthUTC(Number day)

Sets the day of the month to a specified value in the UTC time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>day</td>
<td>Number</td>
<td>The day of month to change to, from 1 to 31. If this value is greater than</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the maximum number of days in the month, the value is set to the last day of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the month.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.setDayOfMonthUTC(9);
gs.info(gdt.getDayOfMonthUTC());
```

Scoped GlideDateTime - setDisplayValue(String asDisplayed)

Sets a date and time value using the current user’s display format and time zone.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>asDisplayed</td>
<td>String</td>
<td>The date and time in the current user's display format and time zone. The parameter must be formatted using the current user's preferred display format, such as MM-dd-yyyy HH:mm:ss. To assign the current date and time to a variable in a workflow script, use variable.setDisplayValue(gs.nowDateTime);</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2014-02-02 12:00:00");
gdt.setDisplayValue("2014-01-01 12:00:00"); // uses current user session time zone (US/Pacific)
gs.info(gdt.getValue());
```

Scoped GlideDateTime - setDisplayValue(String value, String format)

Sets a date and time value using the current user's time zone and the specified date and time format. This method throws a runtime exception if the date and time format used in the value parameter does not match the format parameter. You can retrieve the error message by calling getErrorMsg() on the GlideDateTime object after the exception is caught.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>String</td>
<td>The date and time in the current user's time zone.</td>
</tr>
<tr>
<td>format</td>
<td>String</td>
<td>The date and time format to use to parse the value parameter.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-02-02 12:00:00");
```
gdt.setDisplayValue("20-5-2011 12:00:00", "dd-MM-yyyy HH:mm:ss"); //uses current user session time zone (US/Pacific)
gs.info(gdt.getValue());

Scoped GlideDateTime - setDisplayValueInternal(String value)
Sets a date and time value using the internal format (yyyy-MM-dd HH:mm:ss) and the current user's time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>String</td>
<td>The date and time in internal format.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var gdt = new GlideDateTime("2014-02-02 12:00:00");
gdt.setDisplayValueInternal("2014-01-01 12:00:00"); //uses current user session time zone (US/Pacific)
gs.info(gdt.getValue());

Scoped GlideDateTime - setGlideDateTime(GlideDateTime g)
Sets the date and time of the current object using an existing GlideDateTime object. This method is equivalent to instantiating a new object with a GlideDateTime parameter.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>GlideDateTime</td>
<td>The object to use for setting the datetime value.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var dt1 = new GlideDateTime("2011-01-01 12:00:00");
var dt2 = new GlideDateTime("2011-02-02 08:00:00");
dt1.setGlideDateTime(dt2);
gs.info(dt1.getValue());
Scoped GlideDateTime - setMonthLocalTime(Number month)

Sets the month stored by the GlideDateTime object to the specified value using the current user's time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>month</td>
<td>Number</td>
<td>The month to change to.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.setMonthLocalTime(1);
gs.info(gdt.getMonthLocalTime());
```

Output: 1

Scoped GlideDateTime - setMonthUTC(Number month)

Sets the month stored by the GlideDateTime object to the specified value using the UTC time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>month</td>
<td>Number</td>
<td>The month to change to.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.setMonthUTC(1);
gs.info(gdt.getMonthUTC());
```

Output: 1

Scoped GlideDateTime - setValue(String o)

Sets the date and time of the GlideDateTime object.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| o    | String | The date and time to use. This parameter may be one of several types:  
- A string in the UTC time zone and the internal format of yyyy-MM-dd HH:mm:ss. Sets the value of the object to the specified date and time. Using the method this way is equivalent to instantiating a new GlideDateTime object using the GlideDateTime(String value) constructor. If the date and time format used does not match the internal format, the method attempts to set the date and time using other available formats. Resolving the date and time this way can lead to inaccurate data due to ambiguity in the day and month values. When using a non-standard date and time format, use etValueUTC(String dt, String format) instead.  
- A GlideDateTime object. Sets the value of the object to the date and time stored by the GlideDateTime passed in the parameter. Using the method this way is equivalent to instantiating a new GlideDateTime object using the GlideDateTime(GlideDateTime g) constructor.  
- A JavaScript Number. Sets the value of the object using the Number value as milliseconds past January 1, 1970 00:00:00 GMT. |
Scoped GlideDateTime - setValueUTC(String dt, String format)

Sets a date and time value using the UTC time zone and the specified date and time format. This method throws a runtime exception if the date and time format used in the `dt` parameter does not match the `format` parameter. You can retrieve the error message by calling `getErrorMsg()` on the GlideDateTime object after the exception is caught.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dt</td>
<td>String</td>
<td>The date and time to use.</td>
</tr>
<tr>
<td>format</td>
<td>String</td>
<td>The date and time format to use.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var gdt = new GlideDateTime("2011-01-01 12:00:00");
gdt.setValueUTC("2011-02-02 08:00:00"); // value set = 2011-02-02 08:00:00
gs.info(gdt.getValue());
```

Scoped GlideDateTime - setYearLocalTime(Number year)

Sets the year stored by the GlideDateTime object to the specified value using the current user's time zone.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>Number</td>
<td>The year to change to.</td>
</tr>
</tbody>
</table>

```java
var gdt = new GlideDateTime("2011-01-01 12:00:00");
gdt.setYearLocalTime(2011);
gs.info(gdt.getValue());
```
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.setYearLocalTime(2013);
gs.info(gdt.getYearLocalTime());
```

Output: 2013

#### Scoped GlideDateTime - setYearUTC(Number year)

Sets the year stored by the GlideDateTime object to the specified value using the UTC time zone.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>Number</td>
<td>The year to change to.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime();
gdt.setYearUTC(2013);
gs.info(gdt.getYearUTC());
```

Output: 2013

#### Scoped GlideDateTime - subtract(GlideTime time)

Subtracts a specified amount of time from the current GlideDateTime object.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>GlideTime</td>
<td>The time value to subtract.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
var gtime1 = new GlideTime();
gtime1.setValue("00:00:20");
gdt.subtract(gtime1);
var gtime2 = gdt.getTime();
gs.info(gtime2.getByFormat('hh:mm:ss'));
```

Output: 07:59:40

**ScopedGlideDateTime - subtract(Number milliseconds)**

Subtracts the specified number of milliseconds from the GlideDateTime object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>Number</td>
<td>The number of milliseconds to subtract.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-12-07 08:00:00");
gdt.subtract(1000);
gs.info(gdt.getValue());
```

Output: 2011-12-07 07:59:59

**Scoped GlideDateTime - subtract(GlideDateTime start, GlideDateTime end)**

Gets the duration difference between two GlideDateTime values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>GlideDateTime</td>
<td>The start value.</td>
</tr>
<tr>
<td>End</td>
<td>GlideDateTime</td>
<td>The end value.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GlideDuration</td>
<td>The duration between the two values.</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt1 = new GlideDateTime("2011-08-28 09:00:00");
var gdt2 = new GlideDateTime("2011-08-31 08:00:00");

var dur = GlideDateTime.subtract(gdt1, gdt2); //the difference between gdt1 and gdt2
gs.info(dur.getDisplayValue());
```

Output: 2 Days 23 Hours

**Scoped GlideDateTime - toString()**

Gets the date and time value stored by the GlideDateTime object in the internal format, yyyy-MM-dd HH:mm:ss, and the system time zone, UTC by default. This method is equivalent to getValue().

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.info(gdt.toString());
```

**GlideDBFunctionBuilder**

Build functions to perform SQL operations in the database.

The GlideDBFunctionBuilder methods provide a way to build Relational Database Management System (RDBMS) functions to perform SQL operations on record data. These methods can be used in both scoped and global server scripts.

To use platform functions:

- Construct a function using the GlideDBFunctionBuilder constructor and associated methods.
- After building a function, you can apply the function to the current record using the addFunction() method of the GlideRecord class.
- Add the function to a query using the addQuery() method of the GlideRecord class.
Retrieve the results of the function using the existing GlideRecord API methods such as getValue() and getElement().

Scoped GlideDBFunctionBuilder - add()

Adds the values of two or more integer fields.

Use the field(String field) method to define fields on which the operation is performed.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var functionBuilder = new GlideDBFunctionBuilder();
var myAddingFunction = functionBuilder.add();
myAddingFunction = functionBuilder.field('order');
myAddingFunction = functionBuilder.field('priority');
myAddingFunction = functionBuilder.build();
```

Scoped GlideDBFunctionBuilder - build()

Builds the database function defined by the GlideDBFunctionBuilder object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var functionBuilder = new GlideDBFunctionBuilder();
var myAddingFunction = functionBuilder.add();
myAddingFunction = functionBuilder.field('order');
myAddingFunction = functionBuilder.field('priority');
myAddingFunction = functionBuilder.build();
gs.print(myAddingFunction);
```
Output:

```plaintext
*** Script: glidefunction:add(order,priority)
```

**Scoped GlideDBFunctionBuilder - concat()**

Concatenates the values of two or more fields.

Use the `field(String field)` method to define fields on which the operation is performed.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var functionBuilder = new GlideDBFunctionBuilder();
var myConcatFunction = functionBuilder.concat();
myConcatFunction = functionBuilder.field('short_description');
myConcatFunction = functionBuilder.field('caller_id.name');
myConcatFunction = functionBuilder.build();
```

**Scoped GlideDBFunctionBuilder - constant(String constant)**

Defines a constant value to use in the function. If used with the `dayofweek()` method, the string defines whether to use Sunday or Monday as the first day of the week.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>String</td>
<td>A constant value used in a function. When used with the <code>dayofweek()</code> method, the value defines whether the week starts on a Sunday or Monday.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: Week begins on Sunday.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: Week begins on Monday.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This definition enables the <code>dayofweek()</code> method to return the correct day of the week from a given date. If a value other than 1 or 2 is provided, the <code>dayofweek()</code> method uses Sunday as the first day of the week.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Scoped GlideDBFunctionBuilder - datediff()

Determines the duration using a given start date/time and end date/time.

Use the `field(String field)` method to define start and end date/time fields.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var functionBuilder = new GlideDBFunctionBuilder();
var myDateDiffFunction = functionBuilder.datediff();
myDateDiffFunction = functionBuilder.field('sys_updated_on');
myDateDiffFunction = functionBuilder.field('opened_at');
myDateDiffFunction = functionBuilder.build();
```
Scoped GlideDBFunctionBuilder - dayofweek()

Returns an integer representing the day of the week for a given date.

Use the `field(String field)` method to define the given date/time. Use the `constant(String constant)` method to define whether the week starts on a Sunday or Monday.

This method can be used with MySQL, Oracle, and Microsoft SQL Server databases only. If using an Oracle database, the NLS_TERRITORY setting must be set to a territory with Sunday as the first day of the week.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Integer | If the first day of the week is set to Sunday in the `constant(String constant)` method, return values are associated with the following days of the week:
  - 1: Sunday
  - 2: Monday
  - 3: Tuesday
  - 4: Wednesday
  - 5: Thursday
  - 6: Friday
  - 7: Saturday

If the first day of the week is set to Monday:
  - 1: Monday
  - 2: Tuesday
  - 3: Wednesday
  - 4: Thursday
  - 5: Friday
  - 6: Saturday
  - 7: Sunday

If a value other than 1 or 2 is provided in the `constant(String constant)` method, the `dayofweek()` method uses Sunday as the first day of the week.|

```javascript
var functionBuilder = new GlideDBFunctionBuilder();
var dayOfWeekFunction = functionBuilder.dayofweek();
dayOfWeekFunction = functionBuilder.field('opened_at');
dayOfWeekFunction = functionBuilder.constant('2');
```
```javascript
dayOfWeekFunction = functionBuilder.build();
var gr = new GlideRecord('incident');
gr.addFunction(dayOfWeekFunction);
gr.query();
while(gr.next())
gs.log(gr.getValue(dayOfWeekFunction));
```

**Scoped GlideDBFunctionBuilder - divide()**

Divides the value of one integer field by another.
Use the `field(String field)` method to define fields on which the operation is performed.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var functionBuilder = new GlideDBFunctionBuilder();
var myDivideFunction = functionBuilder.divide();
myDivideFunction = functionBuilder.field('order');
myDivideFunction = functionBuilder.field('priority');
myDivideFunction = functionBuilder.build();
```

**Scoped GlideDBFunctionBuilder - field(String field)**

Defines a field on which a SQL operation is performed.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>String</td>
<td>The field on which you are performing the SQL operation.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var functionBuilder = new GlideDBFunctionBuilder();
var myAddingFunction = functionBuilder.add();
```
Scoped GlideDBFunctionBuilder - GlideDBFunctionBuilder()

Instantiates a GlideDBFunctionBuilder object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var builder = new GlideDBFunctionBuilder();
```

Scoped GlideDBFunctionBuilder - length()

Determines the number of code units in a field.

Use the `field(String field)` method to define fields on which the operation is performed.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var functionBuilder = new GlideDBFunctionBuilder();
var myLengthFunction = functionBuilder.length();
myLengthFunction = functionBuilder.field('short_description');
myLengthFunction = functionBuilder.build();
```

Scoped GlideDBFunctionBuilder - multiply()

Multiplies the values of two integer fields.

Use the `field(String field)` method to define fields on which the operation is performed.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var functionBuilder = new GlideDBFunctionBuilder();
var myMultiplyFunction = functionBuilder.multiply();
myMultiplyFunction = functionBuilder.field('order');
myMultiplyFunction = functionBuilder.field('priority');
myMultiplyFunction = functionBuilder.build();
```

### Scoped GlideDBFunctionBuilder - subtract()

Subtracts the value of one integer field from another.

Use the `field(String field)` method to define fields on which the operation is performed.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var functionBuilder = new GlideDBFunctionBuilder();
var mySubtractFunction = functionBuilder.subtract();
mySubtractFunction = functionBuilder.field('order');
mySubtractFunction = functionBuilder.field('priority');
mySubtractFunction = functionBuilder.build();
```

### GlideDialogWindow

The GlideDialogWindow API provides methods for displaying a dialog in the current window and frame.

Use these methods in scripts anywhere that you can use a client-side JavaScript. These methods are most often called from a UI action with the Client check box selected.

**Note:** This API has been deprecated, use the GlideModalV3 API instead.

### GlideDialogWindow - adjustBodySize()

Adjusts the body height of a dialog window to be the window height minus the header height.

You typically call this method after calling GlideDialogWindow - setSize().
### GlideDialogWindow - destroy()

Closes the dialog window.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
//Destroy the current dialog window.
GlideDialogWindow.get().destroy();
```

### GlideDialogWindow - GlideDialogWindow(String id, Boolean readOnly, Number width, Number height)

Provides methods for displaying a dialog in the current window and frame.

Use these methods in scripts anywhere that you can use a client-side JavaScript. These methods are most often called from a UI Action with the Client check box selected.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Name of the UI page to load into the dialog window.</td>
</tr>
<tr>
<td>readOnly</td>
<td>Boolean</td>
<td>Optional. Flag that indicates whether the dialog window is read only (true) or read/write (false). Default: false</td>
</tr>
<tr>
<td>width</td>
<td>Number</td>
<td>Optional. Size (in pixels) to set the width of the dialog window.</td>
</tr>
<tr>
<td>height</td>
<td>Number</td>
<td>Optional. Size (in pixels) to set the height of the dialog window.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
// Creates a dialog window
var gdw = new GlideDialogWindow('show_list');

// Creates a read-only dialog window
var gdw = new GlideDialogWindow('show_list', true);

// Creates a dialog window that is 400 pixels wide
var gdw = new GlideDialogWindow('show_list', false, 400);

// Creates a dialog window that is 400 pixels wide and 200 pixels tall
var gdw = new GlideDialogWindow('show_list', false, 400, 200);
```

### GlideDialogWindow - removeCloseDecoration()

Removes the close decoration in the upper right corner of the window.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
//Remove the close decoration from the current dialog window.
GlideDialogWindow.get().removeCloseDecoration();

GlideDialogWindow - render()
Renders the dialog window.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdw = new GlideDialogWindow('show_list');
gdw.setTitle('Test');
gdw.setSize(750,300);
gdw.setPreference('table', 'u_test_list');
gdw.setPreference('title', 'A New Title');
gdw.render();
```

GlideDialogWindow - setPreference(String name, String value)
Sets a given window property to a specified value.
Any window property can be set using this method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The window property to set.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value for the window property.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdw = new GlideDialogWindow('show_list');
gdw.setTitle('Test');
```
GlideDialogWindow - setSize(Number width, Number height)

Sets the size of the dialog window.
If you do not pass width and height parameters, a default size is used.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>Number</td>
<td>The width of the dialog window.</td>
</tr>
<tr>
<td>height</td>
<td>Number</td>
<td>The height of the dialog window.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gdw = new GlideDialogWindow('show_list');
gdw.setSize(750,300);
```

GlideDialogWindow - setTitle(String title)

Sets the title of the dialog window.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td>The title for the current window.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
//var gdw = new GlideDialogWindow('show_list');
gdw.setTitle('test');
```

GlideDigest

The scoped GlideDigest class provides methods for creating a message digest from strings or input streams using MD5, SHA1, or SHA256 hash algorithms.
**Scoped GlideDigest - getMD5Base64(String source)**

Create a message digest from a string using the MD5 algorithm. The output string is in Base64.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>String</td>
<td>The source string.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message digest.</td>
</tr>
</tbody>
</table>

```js
var inputString = "Her molasses flowed slowly down the hill.";
var digest = new GlideDigest();
gs.info(digest.getMD5Base64(inputString));
```

**Scoped GlideDigest - getMD5Base64FromInputStream(GlideScriptableInputStream inputStream)**

Create a message digest from an input stream using the MD5 algorithm. The output string is in Base64.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputStream</td>
<td>GlideScriptableInputStream</td>
<td>The source input stream.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message digest.</td>
</tr>
</tbody>
</table>

```js
var inputStream = new GlideSysAttachment().getContentStream(attachmentSysID);
var digest = new GlideDigest();
gs.info(digest.getMD5Base64FromInputStream(inputStream));
```

**Scoped GlideDigest - getMD5Hex(String source)**

Create a message digest from a string using the MD5 algorithm. The output string is in hexadecimal.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>String</td>
<td>The source string.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message digest.</td>
</tr>
</tbody>
</table>

```javascript
var inputString = "Her molasses flowed slowly down the hill.";
var digest = new GlideDigest();
gs.info(digest.getMD5Hex(inputString));
```

Scoped GlideDigest - getMD5HexFromInputStream(GlideScriptableInputStream inputStream)

Create a message digest from an input stream using the MD5 algorithm. The output string is in hexadecimal.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputStream</td>
<td>GlideScriptableInputStream</td>
<td>The source input stream.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message digest.</td>
</tr>
</tbody>
</table>

```javascript
var inputStream = new GlideSysAttachment().getContentStream(attachmentSysID);
var digest = new GlideDigest();
gs.info(digest.getMD5HexFromInputStream(inputStream));
```

Scoped GlideDigest - getSHA1Base64(String source)

Create a message digest from a string using the SHA1 algorithm. The output string is in Base64.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>String</td>
<td>The source string.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>inputStream</td>
<td>GlideScriptableInputStream</td>
<td>The source input stream.</td>
</tr>
</tbody>
</table>

## Scoped GlideDigest - getSHA1Base64FromInputStream( GlideScriptableInputStream inputStream)
Create a message digest from an input stream using the SHA1 algorithm. The output string is in Base64.

```
var inputStream = new GlideSysAttachment().getContentStream(attachmentSysID);
var digest = new GlideDigest();
gs.info(digest.getSHA1Base64FromInputStream(inputStream));
```

## Scoped GlideDigest - getSHA1Hex(String source)
Create a message digest from a string using the SHA1 algorithm. The output string is in hexadecimal.

```
var inputString = "Her molasses flowed slowly down the hill.";
var digest = new GlideDigest();
gs.info(digest.getSHA1Base64(inputString));
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>String</td>
<td>The source string.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message digest.</td>
</tr>
</tbody>
</table>

```javascript
var inputString = "Her molasses flowed slowly down the hill.";
var digest = new GlideDigest();
gs.info(digest.getSHA1Hex(inputString));
```

**Scoped GlideDigest - getSHA1HexFromInputStream( GlideScriptableInputStream inputStream)**

Create a message digest from an input stream using the SHA1 algorithm. The output string is in hexadecimal.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputStream</td>
<td>GlideScriptableInputStream</td>
<td>The source input stream.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message digest.</td>
</tr>
</tbody>
</table>

```javascript
var inputStream = new GlideSysAttachment().getContentStream(attachmentSysID);
var digest = new GlideDigest();
gs.info(digest.getSHA1HexFromInputStream(inputStream));
```

**Scoped GlideDigest - getSHA256Base64(String source)**

Create a message digest from a string using the SHA256 algorithm. The output string is in Base64.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>String</td>
<td>The source string.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message digest.</td>
</tr>
</tbody>
</table>

```javascript
var inputString = "Her molasses flowed slowly down the hill.";
var digest = new GlideDigest();
gs.info(digest.getSHA256Base64(inputString));
```

**Scoped GlideDigest - getSHA256Base64FromInputStream**

Create a message digest from an input stream using the SHA256 algorithm. The output string is in Base64.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputStream</td>
<td>GlideScriptableInputStream</td>
<td>The source input stream.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message digest.</td>
</tr>
</tbody>
</table>

```javascript
var inputStream = new GlideSysAttachment().getContentStream(attachmentSysID);
var digest = new GlideDigest();
gs.info(digest.getSHA256Base64FromInputStream(inputStream));
```

**Scoped GlideDigest - getSHA256Hex**

Create a message digest from a string using the SHA256 algorithm. The output string is in hexadecimal.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>String</td>
<td>The source string.</td>
</tr>
</tbody>
</table>

```javascript
var inputString = new GlideSysAttachment().getContentStream(attachmentSysID);
var digest = new GlideDigest();
gs.info(digest.getSHA256Hex(inputString));
```
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message digest.</td>
</tr>
</tbody>
</table>

```
var inputString = "Her molasses flowed slowly down the hill.";
var digest = new GlideDigest();
gs.info(digest.getSHA256Hex(inputString));
```

**Scoped GlideDigest - getSHA256HexFromInputStream(GlideScriptableInputStream inputStream)**

Create a message digest from an input stream using the SHA256 algorithm. The output string is in hexadecimal.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputStream</td>
<td>GlideScriptableInputStream</td>
<td>The source input stream.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message digest.</td>
</tr>
</tbody>
</table>

```
var inputStream = new GlideSysAttachment().getContentStream(attachmentSysID);
var digest = new GlideDigest();
gs.info(digest.getSHA256HexFromInputStream(inputStream));
```

**Scoped GlideDigest - GlideDigest()**

Creates an instance of scoped GlideDigest.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**GlideDocumentV3**

The GlideDocument class provides the ability to search a DOM element, a document, or a JQuery element.

The GlideDocumentV3 API can be used in client-side scripts using ListV2 and ListV3 APIs. The GlideDocument APIs are accessed using the g_document global object.
GlideDocumentV3 - getElement(String selector, Element context)

Returns a node found in the specified DOM based context or created from the HTML context.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selector</td>
<td>String or Object</td>
<td>The selector expression</td>
</tr>
<tr>
<td>context</td>
<td>String or Object</td>
<td>(Optional) A DOM Element, document, or JQuery object to be searched.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node that matches the selector.</td>
</tr>
</tbody>
</table>

GlideDocumentV3 - getElements(String selector, Element context)

Returns a node list found in the specified DOM based context or created if an HTML context is specified.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selector</td>
<td>String or Object</td>
<td>The selector expression</td>
</tr>
<tr>
<td>context</td>
<td>String or Object</td>
<td>(Optional) A DOM Element, document, or JQuery object to be searched.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node list</td>
<td>A list of nodes that matches the selector.</td>
</tr>
</tbody>
</table>

GlideDuration

The scoped GlideDuration class provides methods for working with spans of time or durations. GlideDuration objects store the duration as a date and time from January 1, 1970, 00:00:00. As a result, setValue() and getValue() use the scoped GlideDateTime object for parameters and return values.

**Scoped GlideDuration - add(GlideDuration duration)**

Add the specified duration to the object.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>GlideDuration</td>
<td>The value to add to the object.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDuration</td>
<td>The sum of the current and the added duration.</td>
</tr>
</tbody>
</table>

```javascript
var duration = new GlideDuration('3 12:00:00');
var duration2 = new GlideDuration('3:00:00');
var answer = duration.add(duration2);
gs.info(answer.getDisplayValue());
```

### Scoped GlideDuration - getByFormat(String format)

Gets the duration in the specified format.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>format</td>
<td>String</td>
<td>The duration format.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current duration in the specified format.</td>
</tr>
</tbody>
</table>

```javascript
var dur = new GlideDuration('3 22:00:00');
gs.info(dur.getByFormat('HH:mm'));
```

### Scoped GlideDuration - getDayPart()

Gets the number of days.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
```
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of days.</td>
</tr>
</tbody>
</table>

```javascript
var dur = new GlideDuration('3 12:00:00');
gs.info(dur.getDayPart());
```

**Scoped GlideDuration - getDisplayValue()**

Gets the display value of the duration in number of days, hours, and minutes.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The number of days, hours, and minutes.</td>
</tr>
</tbody>
</table>

```javascript
var dur = new GlideDuration('3 12:00:00');
gs.info(dur.getDisplayValue());
```

**Scoped GlideDuration - GlideDuration()**

Instantiates a GlideDuration object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Scoped GlideDuration - getDurationValue()**

Gets the duration value in "d HH:mm:ss" format.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Duration value in 24-hour format.</td>
</tr>
</tbody>
</table>

```javascript
var dur = new GlideDuration('3 12:00:00');
gs.info(dur.getDurationValue());
```

Scoped GlideDuration - getRoundedDayPart()

Gets the rounded number of days. If the time part is more than 12 hours, the return value is rounded up. Otherwise, it is rounded down.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The day part, rounded.</td>
</tr>
</tbody>
</table>

```javascript
var dur = new GlideDuration('3 11:00:00');
gs.info(dur.getRoundedDayPart());
```

Scoped GlideDuration - getValue()

Gets the internal value of the GlideDuration object.

GlideDuration objects store the duration as a date and time from January 1, 1970, 00:00:00.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The duration in the object’s internal format, which is the date and time from January 1, 1970, 00:00:00.</td>
</tr>
</tbody>
</table>

```javascript
var dur = new GlideDuration('3 12:00:00');
gs.info(dur.getValue());
```

Output:

1970-01-04 12:00:00

### Scoped GlideDuration - GlideDuration(GlideDuration another)

Instantiates a GlideDuration object by cloning the value of another GlideDuration object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>another</td>
<td>GlideDuration</td>
<td>Another scoped GlideDuration object.</td>
</tr>
</tbody>
</table>

### Scoped GlideDuration - GlideDuration(Number milliseconds)

Instantiates a GlideDuration object with the specified duration.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>Number</td>
<td>The duration value in milliseconds.</td>
</tr>
</tbody>
</table>

### Scoped GlideDuration - GlideDuration(String displayValue)

Instantiates a GlideDuration object with the specified display value.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>displayValue</td>
<td>String</td>
<td>The display value.</td>
</tr>
</tbody>
</table>

### Scoped GlideDuration - setDisplayValue(String asDisplayed)

Sets the display value.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>asDisplayed</td>
<td>String</td>
<td>The duration in &quot;d HH:mm:ss&quot; format.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var dur = new GlideDuration();
dur.setDisplayValue('3 08:00:00');
gs.info(dur.getDisplayValue());
```

**Scoped GlideDuration - setValue(Object o)**

Sets the internal value of the GlideDuration object.

GlideDuration objects store the duration as a date and time from January 1, 1970, 00:00:00.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Object</td>
<td>The duration in the object's internal format, which is the date and time from January 1, 1970, 00:00:00.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var dur = new GlideDuration();
dur.setValue('1970-01-05 08:00:00'); // sets internal DateTime value. The String will be parsed into a GlideDateTime object.
gs.info(dur.getDisplayValue());
```

Output:

4 Days 8 Hours

**Scoped GlideDuration - subtract(GlideDuration duration)**

Subtracts the specified duration from the current duration.
### GlideDuration

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>GlideDuration</td>
<td>Duration to subtract.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDuration</td>
<td>GlideDuration object which includes the updated duration.</td>
</tr>
</tbody>
</table>

```javascript
var duration = new GlideDuration('3 12:00:00');
var duration2 = new GlideDuration('3:00:00');
var answer = duration.subtract(duration2);
gs.info(answer.getDisplayValue());
```

**Output:**

3 Days 9 Hours

### GlideElement

The GlideElement API provides a number of convenient script methods for dealing with fields and their values. GlideElement methods are available for the fields of the current GlideRecord.

#### GlideElement - canCreate()

Determines if the user's role permits the creation of new records in this field.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field can be created, false otherwise.</td>
</tr>
</tbody>
</table>

### Scoped equivalent

To use the `canCreate()` method in a scoped application, use the corresponding scoped method: `canCreate()`.

### GlideElement - canRead()

Determines whether the user's role permits them to read the associated GlideRecord.
### GlideElement - canRead()

Determines whether the field can be read.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field can be read, false otherwise.</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `canRead()` method in a scoped application, use the corresponding scoped method: `canRead()`.

### GlideElement - canWrite()

Determines whether the user's role permits them to write to the associated GlideRecord.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user can write to the field, false otherwise.</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `canWrite()` method in a scoped application, use the corresponding scoped method: `canWrite()`.

### GlideElement - changes()

Determines if the current field has been modified. This functionality is available for all available data types, except Journal fields.

**Note:** The `changes()` method is not supported within ACL scripts.

**Note:** If the GlideRecord on which you are performing this method has only been initialized and read, and has not been written, the underlying before-and-after values are the same. In this case, the method returns "false", as there has been no change to the data store.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field has changed, false otherwise.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `changes()` method in a scoped application, use the corresponding scoped method: `changes()`.  

**GlideElement - changesFrom(Object value)**

Determines if the previous value of the current field matches the specified object.

**Note:** If the GlideRecord on which you are performing this method has only been initialized and read, and has not been written, the underlying before-and-after values are the same. In this case, the method returns "false", as there has been no change to the data store.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Object</td>
<td>An object value to check against the previous value of the current field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the previous value matches the parameter,</td>
</tr>
<tr>
<td></td>
<td>false if it does not.</td>
</tr>
</tbody>
</table>

```java
if (theState.changesTo(resolvedState)) {
    operation = 4; //Resolved
}
else if (theState.changesTo(closedState)) {
    operation = 11; //Resolution Accepted
}
else if (theState.changesFrom(resolvedState) ||
    theState.changesFrom(closedState)) {
    operation = 10; //Re-open
}
else {
    operation = 6; //Update
}
```
Scoped equivalent

To use the `changesFrom()` method in a scoped application, use the corresponding scoped method: `changesFrom()`.

GlideElement - `changesTo(Object value)`

Determines if the new value of a field, after a change, matches the specified object.

**Note:** The `changesTo()` method is not supported within ACL scripts.

**Note:** If the GlideRecord on which you are performing this method has only been initialized and read, and has not been written, the underlying before-and-after values are the same. In this case, the method returns “false”, as there has been no change to the data store.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Object</td>
<td>An object value to check against the new value of the current field.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the new value matches the parameter, false if it does not.</td>
</tr>
</tbody>
</table>

```java
if (theState.changesTo(resolvedState)) {
    operation = 4; //Resolved
} else if (theState.changesTo(closedState)) {
    operation = 11; //Resolution Accepted
} else if (theState.changesFrom(resolvedState) ||
    theState.changesFrom(closedState)) {
    operation = 10; //Re-open
} else {
    operation = 6; //Update
}
```
Scoped equivalent

To use the `changesTo()` method in a scoped application, use the corresponding scoped method: `changesTo()`.

**GlideElement - dateNumericValue()**

Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT for a duration field. Does not require the creation of a GlideDateTime object because the duration field is already a GlideDateTime object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Number of milliseconds since January 1, 1970, 00:00:00 GMT.</td>
</tr>
</tbody>
</table>

**Example**

```javascript
var inc = new GlideRecord('incident');
inc.get('17c90efb13418700cc36b1422244b05d');
gs.info(inc.calendar_duration.dateNumericValue());
```

Output: 98000

Scoped equivalent

To use the `dateNumericValue()` method in a scoped application, use the corresponding scoped method: `dateNumericValue()`.

**GlideElement - debug(Object o)**

Debugs the object and adds debug messages using `setError(String)`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Object</td>
<td>An object to debug.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
**GlideElement - getAttribute(String attributeName)**

Returns the value of the specified attribute from the dictionary. If the attribute is a boolean attribute, use `getBooleanAttribute(String)` to get the value as a boolean rather than as a string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributeName</td>
<td>String</td>
<td>Attribute name</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Attribute value</td>
</tr>
</tbody>
</table>

```javascript
doit();
function doit() {
  var gr = new GlideRecord('sys_user');
  gr.query("user_name","admin");
  if (gr.next()) {
    gs.print("we got one");
    gs.print(gr.location.getAttribute("tree_picker"));
  }
}
```

**Scoped equivalent**

To use the `getAttribute()` method in a scoped application, use the corresponding scoped method: `getAttribute0`.

**GlideElement - getBaseTableName()**

Gets the base table of the field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Name of the base table. This may be different from the table that the field is defined on. See &quot;Tables and Classes&quot; in the product documentation.</td>
</tr>
</tbody>
</table>

### GlideElement - getBooleanAttribute(String attributeName)

Returns the Boolean value of the specified attribute from the dictionary.

To get the value as a string, use `getAttribute(string)`.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributeName</td>
<td>String</td>
<td>Attribute name</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Boolean value of the attribute. Returns false if the attribute does not exist.</td>
</tr>
</tbody>
</table>

### Scoped equivalent

To use the `getBooleanAttribute()` method in a scoped application, use the corresponding scoped method: `getBooleanAttribute0`.

### GlideElement - getChoices(String value)

Generates a choice list for a field. Returns the choice values from the base table only, not from the extended table.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>String</td>
<td>An optional dependent value.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array list</td>
<td>The choice values for the field.</td>
</tr>
</tbody>
</table>

```
var field = gr.getElement('os');
```
var choices = field.getChoices();

Scoped equivalent
To use the `getChoices()` method in a scoped application, use the corresponding scoped method: `getChoices()`.

**GlideElement - `getChoiceValue()`**
Returns the choice label for the current choice value.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The choice label.</td>
</tr>
</tbody>
</table>

Scoped equivalent
To use the `getChoiceValue()` method in a scoped application, use the corresponding scoped method: `getChoiceValue()`.

**GlideElement - `getDebugCount()`**
Returns the number of debug messages logged by `debug()`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of debug messages.</td>
</tr>
</tbody>
</table>

**GlideElement - `getDependent()`**
Checks whether or not the field is dependent on another field.
### GlideElement - getDependentTable()

Returns the table that the current table depends on.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Name of the field on which the current field depends.</td>
</tr>
</tbody>
</table>

### GlideElement - getDisplayValue(Number maxChar)

Returns the formatted display value of the field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxChar</td>
<td>Number</td>
<td>Optional, maximum number of characters to return.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Display value of the field.</td>
</tr>
</tbody>
</table>

```javascript
define('fields', current.getFields());
define('for (var i = 0; i < fields.size(); i++) {
    var field = fields.get(i);
    var name = field.getName();
    var value = field.getDisplayValue();
```
Scoped equivalent

To use the `getDisplayValue()` method in a scoped application, use the corresponding scoped method: `getDisplayValue()`.

GlideElement - `getDisplayValueExt(Number maxChar, String nullSub)`

Returns the formatted display value of a field, or a specified substitute value if the display value is null or empty.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxChar</td>
<td>Number</td>
<td>Optional, the maximum number of characters to be returned.</td>
</tr>
<tr>
<td>nullSub</td>
<td>String</td>
<td>The value to return if the display value is null or empty.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The formatted display value of the field, or the specified substitute value.</td>
</tr>
</tbody>
</table>

GlideElement - `getED()`

Returns an element descriptor.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElementDescriptor</td>
<td>The field's element descriptor.</td>
</tr>
</tbody>
</table>
Scoped equivalent

To use the `getED()` method in a scoped application, use the corresponding scoped method: `getEDO`.

**GlideElement - getElementValue(String value)**

Returns the value for a given element.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>value</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the element</td>
</tr>
</tbody>
</table>

**GlideElement - getError()**

Returns error debug messages.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A string of debug messages</td>
</tr>
</tbody>
</table>

**GlideElement - getEscapedValue()**

Returns the escaped value for the current element.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The escaped value of the current element.</td>
</tr>
</tbody>
</table>

GlideElement - getFieldStyle()

Returns the CSS style for the field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The CSS style for the field.</td>
</tr>
</tbody>
</table>

```javascript
var fields = current.getFields();
for (var i = 0; i < fields.size(); i++) {
    var field = fields.get(i);
    var name = field.getName();
    var value = field.getDisplayValue();
    gs.print(i + ". " + name + ": " + value);
}
```

GlideElement - getGlideObject()

Returns a Glide object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>A Glide object.</td>
</tr>
</tbody>
</table>

```javascript
function calcDateDelta(start, end, calendar) {
    var cal = GlideCalendar.getCalendar(calendar);
    if (!cal.isValid())
        return null;
    var realStart = start.getGlideObject();
}
```
```javascript
var realEnd = end.getGlideObject();
var duration = cal.subtract(realStart, realEnd);
return duration;
}
```

**GlideElement - getGlideRecord()**

Returns a Glide record.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>A glide record</td>
</tr>
</tbody>
</table>

```javascript
function task_ciAllocate() {
  var cnt = g_list.getRowCount();
  if (cnt == 0)
    return;

  var pct = 100.0 / cnt;
  var pct = (parseInt((pct + .005) * 100)) / 100;
  var gr = g_list.getGlideRecord();
  gr.query();
  while (gr.next()) {
    gr.u_allocation = pct;
    gr.update();
  }
}
```

**GlideElement - getHTMLValue(Number maxChars)**

Returns the HTML value of a field.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxChars</td>
<td>Number</td>
<td>Optional. Maximum number of characters to return.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>HTML value for the field.</td>
</tr>
</tbody>
</table>
Scoped equivalent

To use the `getHTMLValue()` method in a scoped application, use the corresponding scoped method: `getHTMLValueO`.

**GlideElement - getHTMLValueExt(Number maxChar, String nullSub)**

Returns the HTML value of a field, or a specified substitute value if the HTML value is null or empty.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxChar</td>
<td>Number</td>
<td>The maximum number of characters to return.</td>
</tr>
<tr>
<td>nullSub</td>
<td>String</td>
<td>The value to return if the HTML value is null or empty.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The HTML value or the specified substitute value.</td>
</tr>
</tbody>
</table>

**GlideElement - getJournalEntry(Number mostRecent)**

Returns either the most recent journal entry or all journal entries.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mostRecent</td>
<td>Number</td>
<td>If 1, returns the most recent entry. If -1, returns all journal entries.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>For the most recent entry, returns a string that contains the field label, timestamp, and user display name of the journal entry. For all journal entries, returns the same information for all journal entries ever entered as a single string with each entry delimited by &quot;\n\n&quot;.</td>
</tr>
</tbody>
</table>

```javascript
// gets all journal entries as a string where each entry is delimited by '\n\n'
var notes = current.work_notes.getJournalEntry(-1);
// stores each entry into an array of strings
var na = notes.split("\n\n");
for (var i = 0; i < na.length; i++)
gs.print(na[i]);
```

Scoped equivalent

To use the `getJournalEntry()` method in a scoped application, use the corresponding scoped method: `getJournalEntry()`.

**GlideElement - `getLabel()`**

Returns the object's label.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Object's label</td>
</tr>
</tbody>
</table>

**Example**

```javascript
var gr = new GlideRecord("sc_req_item");
gr.addQuery("request", current.sysapproval);
gr.query();
while(gr.next()) {
  var nicePrice = gr.price.toString();
  if (nicePrice !== ) {
    nicePrice = parseFloat(nicePrice);
    nicePrice = nicePrice.toFixed(2);
```
Scoped equivalent

To use the `getLabel()` method in a scoped application, use the corresponding scoped method: `getLabel()`.

GlideElement - `getName()`

Returns the name of the field.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Field name</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `getName()` method in a scoped application, use the corresponding scoped method: `getName()`.

GlideElement - `getRefRecord()`

Returns a GlideRecord object for a given reference element.

Warning: If the reference element does not contain a value, it returns an empty GlideRecord object, not a NULL object.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>A GlideRecord object</td>
</tr>
</tbody>
</table>

```javascript
var grINC = new GlideRecord('incident');
grINC.notNullQuery('caller_id');
grINC.query();
if (grINC.next()) {
    // Get a GlideRecord object for the referenced sys_user record
    var grUSER = grINC.caller_id.getRefRecord();
    if (grUSER.isValidRecord())
        gs.print( grUSER.getValue('name') );
}
```

Scoped equivalent

To use the `getRefRecord()` method in a scoped application, use the corresponding scoped method: `getRefRecord()`.

GlideElement - getStyle()

Returns the CSS style for the value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The CSS style for the value.</td>
</tr>
</tbody>
</table>

```javascript
// Get string of style field from Field Style record
var cssStyle = gr.state.getStyle();
```
GlideElement - getTableName()

Returns the name of the field's table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Name of the table. This may be different from the table Class that the record is in. See Tables and Classes in the product documentation.</td>
</tr>
</tbody>
</table>

```java
if (current.approver.getTableName() == "sysapproval_approver") {
    if (current.approver == email.from_sys_id) {
        current.comments = "reply from: " + email.from + "\n\n" +
        email.body_text;

        // if it's been cancelled, it's cancelled.
        var doit = true;
        if (current.state=='cancelled')
            doit = false;

        if (email.body.state != undefined)
            current.state= email.body.state;

        if (doit)
            current.update();
    } else {
        gs.log("Approval for task "+
            "+current.sysapproval.getDisplayValue()+") rejected because
        user sending
        email("+email.from+") does not match the approver
        ("+current.approver.getDisplayValue()+");
    }
}
```

Scoped equivalent

To use the `getTableName()` method in a scoped application, use the corresponding scoped method: `getTableName()`.

GlideElement - getTextAreaDisplayValue()

Returns the value and escapes the HTML.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The escaped HTML</td>
</tr>
</tbody>
</table>

**GlideElement - getXMLValue()**

Returns the XML value of a field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The XML value</td>
</tr>
</tbody>
</table>

**GlideElement - hasAttribute(String attributeName)**

Determines whether a field has a particular attribute.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributeName</td>
<td>String</td>
<td>The attribute to check for</td>
</tr>
</tbody>
</table>
var totalCritical = 0;
var filledCritical = 0; var fields = current.getFields();
gs.print(fields); for (var num = 0; num < fields.size(); num ++)
{
    gs.print("RUNNING ARRAY VALUE " + num);
    var ed = fields.get(num).getED();
    if(ed.hasAttribute("tiaa_critical"))
    {
        gs.print("CRITICAL FIELD FOUND");
        totalCritical ++;
        if (!fields.get(num).isNil()){
            filledCritical ++;
        }
    }
}
var answer = 0; gs.print("TOTAL - " + totalCritical);
gs.print("FILLED - " + filledCritical); if (filledCritical &gt; 0 &amp;&amp; totalCritical &gt; 0){
    var pcnt = (filledCritical/totalCritical)*100;
    answer = pcnt.toFixed(2);;
}
answer;

GlideElement - hasRightsTo(String operationName)
Determines if the user has the right to perform a particular operation.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operationName</td>
<td>String</td>
<td>Name of the operation to check for</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user has permission to perform the operation, false otherwise.</td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field has a value, false otherwise.</td>
</tr>
</tbody>
</table>

**GlideElement - nil()**

Determines whether the field is null.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field is null or an empty string, false otherwise.</td>
</tr>
</tbody>
</table>

```java
if (current.start_date.changes() || current.end_date.changes() || current.assigned_to.changes()) {
    if (!current.start_date.nil() && !current.end_date.nil() && !current.assigned_to.nil()) {
        gs.eventQueue("change.calendar.notify", current, current.assigned_to, previous.assigned_to);
    }
}
```

**Scoped equivalent**

To use the nil() method in a scoped application, use the corresponding scoped method: nil().

**GlideElement - setDateNumericValue(Number milliseconds)**

Sets the duration field to a number of milliseconds since January 1, 1970, 00:00:00 GMT for a duration field. Does not require the creation of a GlideDateTime object because the duration field is already a GlideDateTime object.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>Number</td>
<td>Number of milliseconds spanned by the duration.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### Example

```javascript
var inc = new GlideRecord('incident');
inc.get('17c90efb13418700cc36b1422244b05d');
var timems = inc.calendar_duration.dateNumericValue();
timems = timems + 11*1000;
inc.calendar_duration.setDateNumericValue(timems)
gs.info(inc.calendar_duration.getValue());
```

Output: 1970-01-01 00:01:38

### Scoped equivalent

To use the `setDateNumericValue()` method in a scoped application, use the corresponding scoped method: `setDateNumericValue()`.

#### GlideElement - `setDisplayValue(Object displayValue)`

Sets the display value of the field.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>displayValue</td>
<td>Object</td>
<td>Value to be displayed.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
current.assignment_group.setDisplayValue('Network');
```
Scoped equivalent

To use the `setDisplayValue()` method in a scoped application, use the corresponding scoped method: `setDisplayValue()`.

GlideElement - setError(String message)

Adds an error message.
Can be retrieved using `getError()`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
if (!current.u_date1.nil()) && (!current.u_date2.nil())) {
  var start =
  current.u_date1.getGlideObject().getNumericValue();
  var end = current.u_date2.getGlideObject().getNumericValue();
  if (start > end) {
    gs.addInfoMessage('start must be before end');
    current.u_date1.setError('start must be before end');
    current.setAbortAction(true);
  }
}
```

Scoped equivalent

To use the `setError()` method in a scoped application, use the corresponding scoped method: `setError()`.

GlideElement - setInitialValue(String value)

Sets the initial value of a field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>String</td>
<td>Initial value for the field.</td>
</tr>
</tbody>
</table>
GlideElement - setJournalEntry(Object value, String userName)

Sets the journal entry.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Object</td>
<td>The value to set the journal entry to.</td>
</tr>
<tr>
<td>userName</td>
<td>String</td>
<td>The user to attribute the journal entry to. Does not set the journal entry's created by field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideElement - setValue(Object value)

Sets the value of a field.

Note: Before calling this method, the element must already exist by querying an existing record or by using the `gr.initialize()` method to initialize a new record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Object</td>
<td>The value the field is to be set to.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

// GlideElement.setValue provides same functionality as GlideRecord.setValue
var gr = new GlideRecord('incident');
gr.initialize();
gr.short_description.setValue('A short description.);
gs.info(gr.getElement('short_description'));
Scoped equivalent

To use the `setValue()` method in a scoped application, use the corresponding scoped method: `setValue()`.

**GlideElement - `toString()`**

Converts the field's value to a string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The field's value as a string.</td>
</tr>
</tbody>
</table>

doit();

```javascript
function doit() {
    var gr = new GlideRecord('sys_user');
gr.query();
while (gr.next()) {
    if ((gr.first_name.toString().length !=
gr.first_name.toString().trim().length) ||
    (gr.last_name.toString().length
        != gr.last_name.toString().trim().length)) {
        gr.first_name = gr.first_name.toString().trim();
gr.last_name = gr.last_name.toString().trim();
gr.autoSysFields(false);
gr.update();
    }
}
```

 Scoped equivalent

To use the `toString()` method in a scoped application, use the corresponding scoped method: `toString0`.

**GlideElement**

The Scoped GlideElement API provides a number of convenient script methods for dealing with fields and their values. Scoped GlideElement methods are available for the fields of the current GlideRecord.
**Scoped GlideElement - canCreate()**

Determines if the user's role permits the creation of new records in this field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field can be created, false otherwise.</td>
</tr>
</tbody>
</table>

**Scoped GlideElement - canRead()**

Indicates whether the user's role permits them to read the associated GlideRecord.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field can be read, false otherwise.</td>
</tr>
</tbody>
</table>

**Scoped GlideElement - canWrite()**

Determines whether the user's role permits them to write to the associated GlideRecord.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user can write to the field, false otherwise.</td>
</tr>
</tbody>
</table>

**Scoped GlideElement - changes()**

Determines if the current field has been modified. This functionality is available for all available data types, except Journal fields.
**Note:** The `changes()` method is not supported within ACL scripts.

**Note:** If the GlideRecord on which you are performing this method has only been initialized and read, and has not been written, the underlying before-and-after values are the same. In this case, the method returns “false”, as there has been no change to the data store.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the fields have been changed, false if the field has not.</td>
</tr>
</tbody>
</table>

```java
// This method is often used in business rules. The following example shows it from a business rule,
// if "assigned_to" field value is changed, create a event in the EventQueue.
if (!current.assigned_to.nil() && current.assigned_to.changes())
{
    gs.eventQueue('incident.assigned', current,
    current.assigned_to.getDisplayValue(),
    previous.assigned_to.getDisplayValue());
}
```

### Scoped GlideElement - changesFrom(Object o)

Determines if the previous value of the current field matches the specified object.

**Note:** If the GlideRecord on which you are performing this method has only been initialized and read, and has not been written, the underlying before-and-after values are the same. In this case, the method returns “false”, as there has been no change to the data store.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Object</td>
<td>An object value to check against the previous value of the current field.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the previous value matches, false if it does not.</td>
</tr>
</tbody>
</table>

// The following example shows that in a business rule, if "active" field is changed from true, // insert a event in the EventQueue.
if (current.active.changesFrom(true)) {
    gs.eventQueue("incident.inactive", current,
    current.incident_state, previous.incident_state);
}

Scoped GlideElement - changesTo(Object o)

Determines if the new value of a field, after a change, matches the specified object.

Note: The changesTo() method is not supported within ACL scripts.

Note: If the GlideRecord on which you are performing this method has only been initialized and read, and has not been written, the underlying before-and-after values are the same. In this case, the method returns "false", as there has been no change to the data store.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Object</td>
<td>An object value to check against the new value of the current field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the previous value matches, false if it does not.</td>
</tr>
</tbody>
</table>

// The following example shows that in a business rule, if "active" field is changed to false, // insert a event in the EventQueue.
if (current.active.changesTo(false)) {
    gs.eventQueue("incident.inactive", current,
    current.incident_state, previous.incident_state);
}
Scoped GlideElement - dateNumericValue()

Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT for a duration field. Does not require the creation of a GlideDateTime object because the duration field is already a GlideDateTime object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Number of milliseconds since January 1, 1970, 00:00:00 GMT.</td>
</tr>
</tbody>
</table>

Example

```javascript
var inc = new GlideRecord('incident');
inc.get('17c90efb13418700cc36b1422244b05d');
gs.info(inc.calendar_duration.dateNumericValue());
```

Output: 98000

Scoped GlideElement - getAttribute(String attributeName)

Returns the value of the specified attribute from the dictionary.

If the attribute is a boolean attribute, use getBooleanAttribute(String) to get the value as a boolean rather than as a string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributeName</td>
<td>String</td>
<td>Attribute name</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Attribute value</td>
</tr>
</tbody>
</table>

doit();
function doit() {
  var gr = new GlideRecord('sys_user');
  gr.query("user_name","admin");
  if (gr.next()) {
    gs.print("we got one");
  }
}
Scoped GlideElement - getBooleanAttribute(String attributeName)

Returns the Boolean value of the specified attribute from the dictionary. To get the value as a string, use getAttribute(string).

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributeName</td>
<td>String</td>
<td>Attribute name</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Boolean value of the attribute. Returns false if the attribute does not exist.</td>
</tr>
</tbody>
</table>

Scoped GlideElement - getChoices(String dependent)

Generates a choice list for a field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependent</td>
<td>String</td>
<td>Optional: a dependent value</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array list of choices.</td>
</tr>
</tbody>
</table>

```javascript
var glideRecord = new GlideRecord('incident');
glideRecord.query('priority','1');
glideRecord.next();

// urgency has choice list: 1 - High, 2 - Medium, 3 - Low, with value: 1, 2, 3
var choices = glideRecord.urgency.getChoices();
gs.info(choices);
```

Scoped GlideElement - getChoiceValue()

Returns the choice label for the current choice.

A choice has a value (number) and a label (string). This method returns the label.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The selected choice's label.</td>
</tr>
</tbody>
</table>

```javascript
var glideRecord = new GlideRecord('incident');
glideRecord.query('priority','1');
glideRecord.next();

// urgency has choice list: 1 - High, 2 - Medium, 3 - Low, with value: 1, 2, 3
var choiceLabel = glideRecord.urgency.getChoiceValue();
gs.info(choiceLabel);
```

Output:

```
1 - High
```

### Scoped GlideElement - getDecryptedValue()

Returns the clear text value for Password (2 way encrypted) fields in scoped applications.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The clear text password.</td>
</tr>
</tbody>
</table>

```javascript
var tablename = 'x_scoped_app_table'
var CI = new GlideRecord(tablename);
CI.addQuery('number', '0001002');
CI.query();
CI.next();

var password = CI.password_field
var decrypted = password.getDecryptedValue();
gs.info(decrypted);
```
Output:

```
x_scoped_app: cleartextpassword
```

**Scoped GlideElement - getDisplayValue(Number maxCharacters)**

Gets the formatted display value of the field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxCharacters</td>
<td>Number</td>
<td>Optional: Maximum characters desired</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The display value of the field</td>
</tr>
</tbody>
</table>

```javascript
var glideRecord = new GlideRecord('incident');
glideRecord.query('priority','1');
glideRecord.next();
gs.info(glideRecord.priority.getDisplayValue());
```

**Scoped GlideElement - getED()**

Returns the field’s element descriptor.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scoped GlideElementDescriptor</td>
<td>The field’s element descriptor.</td>
</tr>
</tbody>
</table>

```javascript
var grInc = new GlideRecord('incident');
grInc.query('priority', '1');
var field = grInc.getElement('priority');
var ed = field.getED();
```
Scoped GlideElement - getGlobalDisplayValue()

Returns the phone number in international format.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scoped GlideElement - getHTMLValue(Number maxChars)

Returns the HTML value of a field.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxChars</td>
<td>Number</td>
<td>Optional. Maximum number of characters to return.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>HTML value for the field.</td>
</tr>
</tbody>
</table>

```javascript
var inccause = new GlideRecord("incident");
inccause.short_description = current.short_description;
inccause.comments = current.comments.getHTMLValue();
inccause.insert();
```

Scoped GlideElement - getJournalEntry(Number mostRecent)

Returns either the most recent journal entry or all journal entries.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mostRecent</td>
<td>Number</td>
<td>If 1, returns the most recent entry. If -1, returns all journal entries.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>HTML value for the field.</td>
</tr>
</tbody>
</table>
ServiceNow    Kingston    Now Platform Custom Business Applications

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>For the most recent entry, returns a string that contains the field label,</td>
</tr>
<tr>
<td></td>
<td>timestamp, and user display name of the journal entry.</td>
</tr>
<tr>
<td></td>
<td>For all journal entries, returns the same information for all journal</td>
</tr>
<tr>
<td></td>
<td>entries ever entered as a single string with each entry delimited by &quot;\n\n&quot;.</td>
</tr>
</tbody>
</table>

```javascript
//gets all journal entries as a string where each entry is
delimited by '\n\n'
var notes = current.work_notes.getJournalEntry(-1);
//stores each entry into an array of strings
var na = notes.split("\n\n");
for (var i = 0; i < na.length; i++)
gs.print(na[i]);
```

Scoped GlideElement - getLabel()

Returns the object label.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Object label</td>
</tr>
</tbody>
</table>

Example

```javascript
var gr = new GlideRecord("sc_req_item");
gr.addQuery("request", current.sysapproval);
gr.query();
while(gr.next()) {
    var nicePrice = gr.price.toString();
    if (nicePrice != ) {
        nicePrice = parseFloat(nicePrice);
        nicePrice = nicePrice.toFixed(2);
    }
    template.print(gr.number + ": " + gr.quantity + " X " +
gr.cat_item.getDisplayValue() + " at $" + nicePrice + " each
\n");
    template.print(" Options:\n");
    for (key in gr.variables) {
        var v = gr.variables[key];
```
if(v.getGlideObject().getQuestion().getLabel() != ) {
    template.space(4);
    template.print('     ' +
    v.getGlideObject().getQuestion().getLabel() + " = " +
    v.getDisplayValue() + "\n";
}

**Scoped GlideElement - getName()**

Returns the name of the field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Field name</td>
</tr>
</tbody>
</table>

**Scoped GlideElement - getReferenceTable()**

Gets the table name for a reference element.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The table name of the reference</td>
</tr>
</tbody>
</table>

var grINC = new GlideRecord('incident');
grINC.query('number','INC0010041'); // record assignment group assigned to "CAB Approval"
if (grINC.next()) {
    // Get the table name
    var tableName = grINC.assignment_group.getReferenceTable();
    gs.info( tableName );
}

**Scoped GlideElement - getRefRecord()**

Returns a GlideRecord object for a given reference element.
Warning: If the reference element does not contain a value, it returns an empty GlideRecord object, not a NULL object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>A GlideRecord object</td>
</tr>
</tbody>
</table>

```javascript
var grINC = new GlideRecord('incident');
grINC.addNotNullQuery('caller_id');
grINC.query();
if (grINC.next()) {
    // Get a GlideRecord object for the referenced sys_user record
    var grUSER = grINC.caller_id.getRefRecord();
    if (grUSER.isValidRecord())
        gs.print(grUSER.getValue('name'));
}
```

Scoped GlideElement - getTableName()

Returns the name of the table on which the field resides.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Name of the table. The returned value may be different from the table Class that the record is in. See Tables and Classes in the product documentation.</td>
</tr>
</tbody>
</table>

```javascript
if (current.approver.getTableName() == "sysapproval_approver") {
    if (current.approver == email.from_sys_id) {
        current.comments = "reply from: " + email.from + "\n\n" + email.body_text;
    }
}
```
// if it's been cancelled, it's cancelled.
var doit = true;
if (current.state=='cancelled')
doit = false;

if (email.body.state != undefined)
current.state = email.body.state;

if (doit)
current.update();
} else {
gs.log("Approval for task "+
"+current.sysapproval.getDisplayValue()+") rejected because user sending
email( "+email.from+") does not match the approver
("+current.approver.getDisplayValue()+");
}

### Scoped GlideElement - nil()

Determines if a field is null.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field is null or an empty string, false if not.</td>
</tr>
</tbody>
</table>

```javascript
var glideRecord = new GlideRecord('incident');
glideRecord.query('priority','1');
glideRecord.next();
gs.info(glideRecord.state.nil());
```

### Scoped GlideElement - setDateNumericValue(Number milliseconds)

Sets the value of a date/time element to the specified number of milliseconds since January 1, 1970 00:00:00 GMT.

When called, setDateNumericValue() automatically creates the necessary GlideDateTime/GlideDate/GlideDuration object, and then sets the element to the specified value.

**Note:** Before calling this method, the element must already exist by querying an existing record or by using the `gr.initialize()` method to initialize a new record.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>Number</td>
<td>Number of milliseconds since 1/1/1970</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var gr = new GlideRecord("incident");
gr.initialize();
gr.opened_at.setDateNumericValue(10000);
```

Scoped GlideElement - `setDisplayValue(Object value)`
Sets the display value of the field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Object</td>
<td>The value to set for the field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var glideRecord = new GlideRecord('incident');
glideRecord.query('priority','1');
glideRecord.next();

//change the urgency to 3
glideRecord.urgency.setDisplayValue('3 - Low');
gs.info(glideRecord.urgency);
```

Scoped GlideElement - `setError(String errorMessage)`
Adds an error message. Available in Fuji patch 3.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>errorMessage</td>
<td>String</td>
<td>The error message.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var glideRecord = new GlideRecord('incident');
glideRecord.query('priority','1');
glideRecord.next();
glideRecord.short_description.setError('Error text');
```

**Scoped GlideElement - setPhoneNumber(Object phoneNumber, Boolean strict)**

Sets the field to the specified phone number.

This method is only available on a phone number GlideElement.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phoneNumber</td>
<td>Object</td>
<td>The phone number to set. This can be in either the international or local format.</td>
</tr>
<tr>
<td>strict</td>
<td>Boolean</td>
<td>When true, specifies that the number specified must match the correct format. When false, the system attempts to correct an improperly formatted phone number.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the value was set.</td>
</tr>
</tbody>
</table>

**Scoped GlideElement - setValue(Object value)**

Sets the value of a field.

**Note:** Before calling this method, the element must already exist by querying an existing record or by using the `gr.initialize()` method to initialize a new record.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Object</td>
<td>Object value to set the field to.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var glideRecord = new GlideRecord('incident');
glideRecord.query('priority','1');
glideRecord.next();
glideRecord.short_description.setValue('Network failure');
gs.info(glideRecord.short_description);
```

Scoped GlideElement - toString()
Converting the value to a string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Object</td>
<td>Object value to set the field to.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value as a string</td>
</tr>
</tbody>
</table>

```javascript
var glideRecord = new GlideRecord('incident');
glideRecord.query('priority','1');
glideRecord.next();
gs.info(glideRecord.opened_at.toString());
```

GlideElementDescriptor
The scoped GlideElementDescriptor API provides information about individual fields.

There is no constructor for this class. Use the GlideElement `getED()` method to obtain a GlideElementDescriptor object.

Scoped GlideElementDescriptor - getAttachmentEncryptionType()

Returns the encryption type used for attachments on the element's table.

This method is for use with the Edge Encryption plugin.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The encryption type used on attachments. Returns null if attachments on the element's table are not being encrypted.</td>
</tr>
</tbody>
</table>

```javascript
var grInc = new GlideRecord('incident');
grInc.query('priority', '1');
var field = grInc.getElement('priority');
var ed = field.getED();
var isEdge = ed.getAttachmentEncryptionType();
gs.info(isEdge);
```

Output: null

**Scoped GlideElementDescriptor - getEncryptionType()**

Returns the element's encryption type.

This method is for use with the Edge Encryption plugin.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The element's encryption type. Returns null if the element is not encrypted.</td>
</tr>
</tbody>
</table>

```javascript
var grInc = new GlideRecord('incident');
grInc.query('priority', '1');
var field = grInc getElement('priority');
var ed = field.getED();
var isEdge = ed.getEncryptionType();
gs.info(isEdge);
```
Scoped GlideElementDescriptor - getInternalType()

Returns the element's internal data type.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The element's internal data type.</td>
</tr>
</tbody>
</table>

```javascript
var grInc = new GlideRecord('incident');
grInc.query('priority', '1');
var field = grInc.getElementById('priority');
var ed = field.getED();
var isEdge = ed.getInternalType();
gs.info(isEdge);
```

Output:

integer

Scoped GlideElementDescriptor - getLabel()

Returns the element's label.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The element's label.</td>
</tr>
</tbody>
</table>

```javascript
var grInc = new GlideRecord('incident');
grInc.query('priority', '1');
var field = grInc.getElementById('priority');
var ed = field.getED();
```
var isEdge = ed.getLabel();
gs.info(isEdge);

Output: Priority

**Scoped GlideElementDescriptor - getLength()**

Returns the element's length.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The element's size.</td>
</tr>
</tbody>
</table>

```javascript
var grInc = new GlideRecord('incident');
grInc.query('priority', '1');
var field = grInc.getElement('priority');
var ed = field.getED();
var isEdge = ed.getLength();
gs.info(isEdge);
```

Output: 40

**Scoped GlideElementDescriptor - getName()**

Returns the element's name.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The element's name.</td>
</tr>
</tbody>
</table>

```javascript
var grInc = new GlideRecord('incident');
grInc.query('priority', '1');
```
var field = grInc.getElement('priority');
var ed = field.getED();

var isEdge = ed.getName();
gs.info(isEdge);

Output: priority

Scoped GlideElementDescriptor - getPlural()

Returns the element's plural label.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The element's plural label.</td>
</tr>
</tbody>
</table>

var gr = new GlideRecord('incident');
gr.query();
var ed = gr.getED();
gs.info(ed.getPlural());

Output: Incidents

Scoped GlideElementDescriptor - hasAttachmentsEncrypted()

Returns true if an encrypted attachment has been added to the table.
This method is for use with the Edge Encryption plugin.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if an encrypted attachment has been added to the table.</td>
</tr>
</tbody>
</table>

var grInc = new GlideRecord('incident');
grInc.query('priority', '1');
var field = grInc getElement('priority');
var ed = field getED();

var isEdge = ed hasAttachmentsEncrypted();
gs.info(isEdge);

Output: false

Scoped GlideElementDescriptor - isAutoOrSysID()

Returns true if the element is an automatically generated or system field. Automatically generated and system fields cannot be encrypted. This method is for use with the Edge Encryption plugin.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the element is automatically generated or a system field.</td>
</tr>
</tbody>
</table>

var grInc = new GlideRecord('incident');
grInc.query('priority', '1');
var field = grInc getElement('priority');
var ed = field getED();

isEdge = ed isAutoOrSysID();
gs.info(isEdge);

Output: false

Scoped GlideElementDescriptor - isChoiceTable()

Returns true if the element is defined as a dropdown choice in its dictionary definition. Choice fields cannot be encrypted.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Boolean</td>
<td>Returns true if the element is defined as a dropdown choice. Returns true even if there are no entries defined in the choice table. The last choice type, suggestion, does not return true.</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var grInc = new GlideRecord('incident');
grInc.query('priority', '1');
var field = grInc.getElementById('priority');
var ed = field.getED();
var isChoiceTable = ed.isChoiceTable();
gs.info(isChoiceTable);
```

Output: true

**Scoped GlideElementDescriptor - isEdgeEncrypted()**

Returns true if an element is encrypted.

This method is for use with the Edge Encryption plugin.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the element is encrypted, false otherwise.</td>
</tr>
</tbody>
</table>

```javascript
var grInc = new GlideRecord('incident');
grInc.query('priority', '1');
var field = grInc.getElementById('priority');
var ed = field.getED();
var isEdge = ed.isEdgeEncrypted();
gs.info(isEdge);
```

Output: false

**Scoped GlideElementDescriptor - isVirtual()**

Returns true if the element is a virtual element.
A virtual element is a calculated field as set by the dictionary definition of the field. Virtual fields cannot be encrypted.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the element is a virtual element.</td>
</tr>
</tbody>
</table>

```javascript
var grInc = new GlideRecord('incident');
grInc.query('priority', '1');

var field = grInc.getElementById('priority');
var ed = field.getED();

var isVirtual = ed.isVirtual();
gs.info(isVirtual);
```

Output: false

### GlideEmailOutbound

The scoped GlideEmailOutbound class implements the email object for scoped applications. You can use the GlideEmailOutbound methods with the email global object available in mail scripts. The email object behaves identically for global and scoped applications.

### Scoped GlideEmailOutbound - addAddress(String type, String address)

Adds the address to either the cc or bcc list.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Either cc or bcc, determines the list to which the address is added.</td>
</tr>
<tr>
<td>address</td>
<td>String</td>
<td>The recipient's email address.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
```java
email.addAddress('cc', 'joe.employee@something.com');
```

**Scoped GlideEmailOutbound - addAddress(String type, String address, String displayName)**

Adds the recipient to either the cc or bcc list, but uses the display name instead of the address when showing the recipient.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Either cc or bcc, determines the list to which the address is added.</td>
</tr>
<tr>
<td>address</td>
<td>String</td>
<td>The recipient's email address.</td>
</tr>
<tr>
<td>displayName</td>
<td>String</td>
<td>The name to be shown instead of the email address.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
email.addAddress('bcc', 'joe.employee@something.com', 'dudley rocks');
```

**Scoped GlideEmailOutbound - getSubject()**

Returns the email's subject line.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The email's subject line.</td>
</tr>
</tbody>
</table>
```javascript
var subject = email.getSubject();

Scoped GlideEmailOutbound - GlideEmailOutbound()

Instantiates a scoped GlideEmailOutbound object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

var e = new GlideEmailOutbound();

Scoped GlideEmailOutbound - setBody(String bodyText)

Sets the body of the email.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bodyText</td>
<td>String</td>
<td>The body of the email.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

e.email.setBody('Dear Sir, ...');

Scoped GlideEmailOutbound - setFrom(String address)

Sets the sender's address.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>String</td>
<td>The sender's email address.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
email.setFrom('joe.employee@something.com');
```

Scoped GlideEmailOutbound - `setReplyTo(String address)`
Sets the reply to address.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>String</td>
<td>The reply to email address.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
email.setReplyTo('joe.employee@something.com');
```

Scoped GlideEmailOutbound - `setSubject(String subject)`
Sets the email's subject line.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject</td>
<td>String</td>
<td>Text for the subject line.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
email.setSubject('Important Issues to discuss');
```
**GlideEncrypter**

GlideEncrypter provides methods to encrypt and decrypt strings using the Triple DES algorithm.

The GlideEncrypter class is used in server scripts in the global scope. The GlideEncrypter class has two constructors:

- GlideEncrypter()
- GlideEncrypter(String key)

**GlideEncrypter - decrypt(String encryptedString)**

Decrypts a clear string using the Triple DES algorithm.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>encryptedString</td>
<td>String</td>
<td>String to be decrypted.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Clear text string.</td>
</tr>
</tbody>
</table>

```javascript
var encr = new GlideEncrypter();
var clearString = 'abcdefg';
var encrString = encr.encrypt(clearString);
var decrString = encr.decrypt(encrString);
gs.print("Decrypted string = " + decrString);
```

Output:

Decrypted string = abcdefg

**GlideEncrypter - encrypt(String clearString)**

Encrypts a clear string using the Triple DES algorithm.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clearString</td>
<td>String</td>
<td>String to be encrypted.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Encrypted string.</td>
</tr>
</tbody>
</table>

```javascript
var encr = new GlideEncrypter();
var clearString = 'abcdefg';
var encrString = encr.encrypt(clearString);
gs.print("Encrypted string = " + encrString);
```

Output:

```
Encrypted string = 3wjpvKtUIi4=
```

**GlideEncrypter - GlideEncrypter()**

Creates an instance of the GlideEncrypter class using a default (static) encryption key.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var encr = new GlideEncrypter();
```

**GlideEncrypter - GlideEncrypter(String key)**

Creates an instance of the GlideEncrypter class using a given encryption key.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>String</td>
<td>Your encryption key must be exactly 24 characters. A key longer than 24 characters will be truncated.</td>
</tr>
</tbody>
</table>

```javascript
var encr = new GlideEncrypter(myKey);
```

**GlideScopedEvaluator**

The GlideScopedEvaluator API allows you to evaluate scripts in a GlideRecord field from both scoped and global server scripts.
The GlideScopedEvaluator API evaluates scripts within the script field type. The scope of the record defines the scope of the script.

**GlideScopedEvaluator - evaluateScript(GlideRecord grObj, String scriptField, Object variables)**

Evaluates a script from a GlideRecord field.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grObj</td>
<td>GlideRecord</td>
<td>The GlideRecord containing a script expression.</td>
</tr>
<tr>
<td>scriptField</td>
<td>String</td>
<td>(Optional) The name of the field containing the script expression.</td>
</tr>
<tr>
<td>variables</td>
<td>Object</td>
<td>(Optional) A map of variables with name-value pairs. These variables are available to the script during execution of this method.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The result of the script execution.</td>
</tr>
</tbody>
</table>

// For this example, we created a table: "u_global_table" with two columns: "short_description", "test_script". "test_script" will store the script to be evaluated by GlideScopedEvaluator.

```javascript
gr = new GlideRecord('u_global_table');
gr.u_short_description = 'Testing GlideScopedEvaluator';
gr.u_test_script = "gs.getUser().getName() + ' says ' + greeting; ";
gr.insert();

// setup variables to be used by the script
var vars = {'greeting' : 'hello'};

//Evaluate the script from the field
var evaluator = new GlideScopedEvaluator();
gr = new GlideRecord('u_global_table');
gr.addQuery('u_short_description','Testing GlideScopedEvaluator');
gr.query();
if (gr.next()) {
    gs.info(evaluator.evaluateScript(gr, 'u_test_script', vars));
}
```

Output: admin says hello
Scoped equivalent

To use the `evaluateScript()` method in a scoped application, use the corresponding scoped method: `evaluateScript()`.

**GlideScopedEvaluator - `getVariable(String name)`**

Returns a variable from a GlideScopedEvaluator object.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the variable.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The value of the specified variable.</td>
</tr>
</tbody>
</table>

```java
//setting up a record that contains the script to be executed.
gr = new GlideRecord('u_global_table');
gr.u_short_description = 'Calculate Addition';
gr.u_test_script = "result = x + y";
gr.insert();

var evaluator = new GlideScopedEvaluator();
evaluator.putVariable('x', 100);
evaluator.putVariable('y', 200);
evaluator.putVariable('result', null);

// Now retrieve the result
gr = new GlideRecord('u_global_table');
gr.addQuery('u_short_description','Calculate Addition');
gr.query();
if (gr.next()) {
    evaluator.evaluateScript(gr, 'u_test_script', null);
    gs.info(evaluator.getVariable('result'));
}
```

Output: 300

Scoped equivalent

To use the `getVariable()` method in a scoped application, use the corresponding scoped method: `getVariable()`.

**GlideScopedEvaluator - `GlideScopedEvaluator()`**

Instantiates a GlideScopedEvaluator object.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the GlideScopedEvaluator() method in a scoped application, use the corresponding scoped method: GlideScopedEvaluator().

GlideScopedEvaluator - putVariable(String name, Object value)

Puts a variable into the GlideScopedEvaluator object. These variables are available to the script that this GlideScopedEvaluator object runs.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the variable.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>The value of the variable.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
//setting up a record that contains the script to be executed.
gr = new GlideRecord('u_global_table');
gr.u_short_description = 'Calculate Addition';
gr.u_test_script = "result = x + y";
gr.insert();

var evaluator = new GlideScopedEvaluator();
evaluator.putVariable('x', 100);
evaluator.putVariable('y', 200);
evaluator.putVariable('result', null);

// Now retrieve the result
gr = new GlideRecord('u_global_table');
gr.addQuery('u_short_description','Calculate Addition');
gr.query();
if (gr.next()) {
    evaluator.evaluateScript(gr, 'u_test_script', null);
    gs.info(evaluator.getVariable('result'));
}
```

Output: 300
**Scoped equivalent**

To use the `putVariable()` method in a scoped application, use the corresponding scoped method: `putVariable()`.

**GlideScopedEvaluator**

The GlideScopedEvaluator API allows you to evaluate scripts in a GlideRecord field from both scoped and global server scripts.

The GlideScopedEvaluator API evaluates records with script fields defined. The scope of the script is defined by the scope of the record.

**GlideScopedEvaluator - evaluateScript(GlideRecord grObj, String scriptField, Object variables)**

Evaluates a script from a GlideRecord field.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grObj</td>
<td>GlideRecord</td>
<td>The GlideRecord containing a script expression.</td>
</tr>
<tr>
<td>scriptField</td>
<td>String</td>
<td>(Optional) The name of the field containing the script expression.</td>
</tr>
<tr>
<td>variables</td>
<td>Object</td>
<td>(Optional) A map of variables with name-value pairs. These variables are available to the script during execution of this method.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The result of the script execution.</td>
</tr>
</tbody>
</table>

```java
// For this example, we created a table: "x_app_table" with two columns: "short_description", "test_script" // "test_script" will store the script to be evaluated by GlideScopedEvaluator.
gr = new GlideRecord('x_app_table');
gr.short_description = 'Testing GlideScopedEvaluator';
gr.test_script = "gs.getUser().getName() + ' says ' + greeting; ";
gr.insert();

// setup variables to be used by the script
var vars = {'greeting' : 'hello'};

//Evaluate the script from the field
var evaluator = new GlideScopedEvaluator();
gr = new GlideRecord('x_app_table');
```
```java
gr.addQuery('short_description','Testing GlideScopedEvaluator');
gr.query();
if (gr.next()) {
    gs.info(evaluator.evaluateScript(gr, 'test_script', vars));
}
```

Output: admin says hello

**GlideScopedEvaluator - getVariable(String name)**

Returns a variable from a GlideScopedEvaluator object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the variable.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The value of the specified variable.</td>
</tr>
</tbody>
</table>

```java
//setting up a record that contains the script to be executed.
gr = new GlideRecord('x_app_table');
gr.short_description = 'Calculate Addition';
gr.calculate = "result = x + y";
gr.insert();

var evaluator = new GlideScopedEvaluator();
evaluator.putVariable('x', 100);
evaluator.putVariable('y', 200);
evaluator.putVariable('result', null);

// Now retrieve the result
gr = new GlideRecord('x_app_table');
gr.addQuery('short_description','Calculate Addition');
gr.query();
if (gr.next()) {
    evaluator.evaluateScript(gr, 'calculate', null);
    gs.info(evaluator.getVariable('result'));
}
```

**GlideScopedEvaluator - GlideScopedEvaluator()**

Instantiates a GlideScopedEvaluator object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GlideScopedEvaluator - putVariable(String name, Object value)

Puts a variable into the GlideScopedEvaluator object. These variables are available to the script that this GlideScopedEvaluator object runs.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the variable.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>The value of the variable.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
//setting up a record that contains the script to be executed.
gr = new GlideRecord('x_app_table');
gr.short_description = 'Calculate Addition';
gr.calculate = "result = x + y";
gr.insert();

var evaluator = new GlideScopedEvaluator();
evaluator.putVariable('x', 100);
evaluator.putVariable('y', 200);
evaluator.putVariable('result', null);

// Now retrieve the result
gr = new GlideRecord('x_app_table');
gr.addQuery('short_description','Calculate Addition');
gr.query();
if (gr.next()) {
    evaluator.evaluateScript(gr, 'calculate', null);
    gs.info(evaluator.getVariable('result'));
}
```

Output: 300

GlideExcelParser

You can parse .xlsx formatted Excel files.

The GlideExcelParser methods can be used in global and scoped scripts. The API name space identifier "sn_impex" must be used when creating a GlideExcelParser object.

GlideExcelParser - close()

Close the connection to the input stream and release the document.
### GlideExcelParser - getColumnHeaders()

Returns a list of column headers from the parsed document.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

| Array | An array of strings of column headers from the parsed document. |

### GlideExcelParser - getErrorMessage()

Returns the error message when the `parse()` method fails.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The error message.</td>
</tr>
</tbody>
</table>

### GlideExcelParser - getRow()

Get the current row values and headers.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.  
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The row headers are property names and the row values are property values.</td>
</tr>
</tbody>
</table>

**GlideExcelParser - GlideExcelParser()**

Creates an instance of GlideExcelParser.

The API name space identifier "sn_impex" must be used when creating a GlideExcelParser object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var parser = new sn_impex.GlideExcelParser();
var attachment = new GlideSysAttachment();
// use attachment sys id of an excel file
var attachmentStream = attachment.getContentStream(<attachment sys id>);

parser.parse(attachmentStream);

//retrieve the column headers
var headers = parser.getColumnHeaders();
var header1 = headers[0];
var header2 = headers[1];

//print headers
gs.print(header1 + ' ' + header2);

while(parser.next()) {
    var row = parser.getRow();
    //print row value for both columns
    gs.print(row[header1] + ' ' + row[header2])
}
```

**GlideExcelParser - next()**

Moves to the next row.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if there is a next row, otherwise, returns false.</td>
</tr>
</tbody>
</table>

#### GlideExcelParser - parse(GlideScriptableInputStream inputStream)

Parse an XLSX formatted Excel document.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputStream</td>
<td>GlideScriptableInputStream</td>
<td>The Excel document to be parsed.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the parse was successful, otherwise, returns false.</td>
</tr>
</tbody>
</table>

Example from a scripted REST API script where the Excel file is sent in the request payload.

```javascript
var parser = new sn_impex.GlideExcelParser();
parser.parse(request.body.dataStream);
```

#### GlideExcelParser - setHeaderRowNumber(Number headerRowNumber)

Set the number of the header row to be retrieved.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headerRowNumber</td>
<td>Number</td>
<td>The header row to be retrieved.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### GlideExcelParser - setNullToEmpty(Boolean empty)

Return an empty value instead of null when an Excel cell is not present.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>empty</td>
<td>Boolean</td>
<td>When true, cells that are not present return an empty value. When false, cells that are not present return null.</td>
</tr>
</tbody>
</table>

### GlideExcelParser - setSheetName(String sheetName)

Set the name of the sheet to be retrieved.

If both `setSheetNumber()` and `setSheetName()` are set, `setSheetName()` is used.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sheetName</td>
<td>String</td>
<td>The name of the sheet to be retrieved.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### GlideExcelParser - setSheetNumber(Number sheetNumber)

Set the number of the Excel sheet to be retrieved.

If both `setSheetNumber()` and `setSheetName()` are set, `setSheetNumber()` is ignored.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sheetNumber</td>
<td>Number</td>
<td>The Excel sheet number to retrieve.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
**GlideFilter**

The Scoped GlideFilter API provides a method to determine if a record meets a specified set of requirements.

There is no constructor for Scoped GlideFilter. It is accessed by using the global object "GlideFilter".

**Scoped GlideFilter - checkRecord(GlideRecord gr, String filter, Boolean match)**

Compares a specified filter to the contents of a specified GlideRecord.

If the specified filter contains one condition, the method returns true if the record meets the condition of the filter.

If the specified filter contains more than one condition, for example "active=true^number=abc^category=request", you can use the match parameter to define whether all conditions must be met to determine a match or just a single condition.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>GlideRecord to evaluate.</td>
</tr>
<tr>
<td>filter</td>
<td>String</td>
<td>Encoded query string (case-sensitive).</td>
</tr>
</tbody>
</table>
| match | Boolean   | Optional. Flag that indicates whether all conditions must be met if the filter parameter contains multiple conditions. Valid values:  
- true: all conditions must be met for the method to return true  
- false: only one of the conditions must be met for the method to return true  
  Default: true |

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Boolean | Results of the filter comparison.  
- true: filter conditions were met  
- false: filter conditions were not met |

```javascript
var rec = new GlideRecord('incident');
rec.query();
var bool = true;
while(rec.next())
```
GlideFilter

```javascript
{  bool = GlideFilter.checkRecord(rec, "active=true");
  gs.info("number "+ rec. number + " is "+ bool);
}
```

GlideForm

The GlideForm API provides methods to customize forms. GlideForm.js is the JavaScript class containing the methods. The global object g_form is used to access GlideForm methods. GlideForm methods are only used on the client.

These methods are used to make custom changes to the form view of records. All validation of examples was done using Client Scripts.

Some of these methods can also be used in other client scripts (such as Catalog Client Scripts or Wizard Client Scripts), but must first be tested to determine whether they will work as expected.

**Note:** The methods `getControl()`, `getHelpTextControl()`, `getElement()`, and `getFormElement()` are deprecated for mobile devices. For information on using GlideForm for mobile, see [Mobile Client GlideForm (g_form) Scripting and Migration](#).

There is no constructor for the GlideForm class. Access GlideForm methods using the g_form global object.

**GlideForm - addDecoration(String fieldName, String icon, String title)**

Adds an icon on a field’s label.

Adding the same item twice is prevented; however, you can add the same icon with a different title.

**Note:** This method is not supported by Service Catalog.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>The font icon to show next to the field. Supported icons - icon-user, icon-user-group, icon-lightbulb, icon-home, icon-mobile, icon-comment, icon-mail, icon-locked, icon-database, icon-book, icon-drawer, icon-folder, icon-catalog, icon-tab, icon-cards, icon-tree-right, icon-tree, icon-book-open, icon-paperclip, icon-edit, icon-trash, icon-image, icon-search, icon-power, icon-cog, icon-star, icon-star-empty, icon-new-ticket, icon-dashboard, icon-cart-full, icon-view, icon-label, icon-filter, icon-calendar, icon-script, icon-add, icon-delete, icon-help, icon-info, icon-check-circle, icon-alert, icon-sort-ascending, icon-console, icon-list, icon-form, and icon-livefeed.</td>
</tr>
<tr>
<td>title</td>
<td>String</td>
<td>The text title for the icon.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
g_form.addDecoration('caller_id', 'icon-star', 'preferred member');
```

**GlideForm - addDecoration(String fieldName, String icon, String title, String color)**

Adds an icon on a field’s label.

Adding the same item twice is prevented; however, you can add the same icon with a different title.

**Note:** This method is not supported by Service Catalog.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>The font icon to show next to the field. Supported icons - icon-user, icon-user-group, icon-lightbulb, icon-home, icon-mobile, icon-comment, icon-mail, icon-locked, icon-database, icon-book, icon-drawer, icon-folder, icon-catalog, icon-tab, icon-cards, icon-tree-right, icon-tree, icon-book-open, icon-paperclip, icon-edit, icon-trash, icon-image, icon-search, icon-power, icon-cog, icon-star, icon-star-empty, icon-new-ticket, icon-dashboard, icon-cart-full, icon-view, icon-label, icon-filter, icon-calendar, icon-script, icon-add, icon-delete, icon-help, icon-info, icon-check-circle, icon-alert, icon-sort-ascending, icon-console, icon-list, icon-form, and icon-livefeed.</td>
</tr>
</tbody>
</table>

| title   | String | The text title for the icon. |
| color   | String | A CSS color. |

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
// Example usage of addDecoration method
g_form.addDecoration('caller_id', 'icon-star', 'Mark as Favorite', 'color-green');
```

**GlideForm - addErrorMessage(String message)**

Displays the error message at the top of the form.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to display.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
g_form.addErrorMessage('This is an error');
```

**GlideForm - addInfoMessage(String message)**

Adds an informational message to the top of the form.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to display.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
g_form.addInfoMessage('The top five fields in this form are mandatory');
```

**GlideForm - addOption(String fieldName, String choiceValue, String choiceLabel)**

Adds a choice to the end of a choice list field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The name of the field.</td>
</tr>
<tr>
<td>choiceValue</td>
<td>String</td>
<td>The value to be stored in the database.</td>
</tr>
<tr>
<td>choiceLabel</td>
<td>String</td>
<td>The value displayed.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
g_form.addOption('priority', '6', '6 - Really Low');

GlideForm - addOption(String fieldName, String choiceValue, String choiceLabel, Number choiceIndex)

Adds a choice to the choice list field at the position specified.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
<tr>
<td>choiceValue</td>
<td>String</td>
<td>The value stored in the database.</td>
</tr>
<tr>
<td>choiceLabel</td>
<td>String</td>
<td>The value displayed.</td>
</tr>
<tr>
<td>choiceIndex</td>
<td>Number</td>
<td>Order of the choice in the list. The index is into a zero based array.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

g_form.addOption('priority', '2.5', '2.5 - Moderately High', 3);

GlideForm - clearMessages()

Removes all informational and error messages from the top of the form.

Removes informational and error messages added with g_form.addInfoMessage() and g_form.addErrorMessage().

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
GlideForm - clearOptions(String fieldName)
Removes all options from the choice list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideForm - clearValue(String fieldName)
Removes any value(s) from the field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideForm - disableAttachments()
Prevents file attachments from being added.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
**GlideForm - enableAttachments()**

Allows file attachments to be added. Shows the paper clip icon.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideForm - flash(String fieldName, String color, Number count)**

Used to draw attention to a particular field. Flashes the specified color for a specified duration of time in the specified field.

This method is not supported by Service Catalog.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| fieldName | String  | Specifies the field to highlight in the following format: 
| color     | String  | RGB color or acceptable CSS color. |
| count     | Number  | Specifies how long the label will flash. Options include: |
|           |         |   · 2: Flashes for 1 second |
|           |         |   · 0: Flashes for 2 seconds |
|           |         |   · -2: Flashes for 3 seconds |
|           |         |   · -4: Flashes for 4 seconds |

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
GlideForm - getActionName()

Returns the most recent action name, or, for a client script, the sys_id of the UI action clicked.

**Note:** Not available in Wizard client scripts.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current action name.</td>
</tr>
</tbody>
</table>

```javascript
function onSubmit() {
    var action = g_form.getActionName();
    alert('You pressed ' + action);
}
```

GlideForm - getBooleanValue(String fieldName)

Returns a Boolean value for the specified field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns false if the field value is false or undefined; otherwise returns true.</td>
</tr>
</tbody>
</table>

GlideForm - getControl(String fieldName)

Returns the HTML element for the specified field.

Compound fields may contain several HTML elements. This method is generally not necessary as there are built-in methods that use the fields on a form.
If the field is a reference field and the control is a choice list, `getControl()` may not return a control as expected. In this case, use `sys_select.<table name>.<field name>`.

This method is not available in mobile scripts or Service Portal scripts.

---

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTMLElement</td>
<td>The field's HTML element.</td>
</tr>
</tbody>
</table>

---

**GlideForm - getDecimalValue(String fieldName)**

Returns the decimal value of the specified field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The name of the field.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The decimal value of the specified field.</td>
</tr>
</tbody>
</table>

```
function onChange(control, oldValue, newValue, isLoading) {
    alert(g_form.getDecimalValue('percent_complete'));
}
```

---

**GlideForm - getElement(String id)**

Returns the HTML element specified by the parameter.

Compound fields may contain several HTML elements. This method is generally not necessary as there are built-in methods that use the fields on a form.

This method is not available in mobile scripts or Service Portal scripts.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>The field id.</td>
</tr>
</tbody>
</table>
**GlideForm - getFormElement()**

Returns the HTML element for the form.

This method is not available in mobile scripts or Service Portal scripts.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTMLFormElement</td>
<td>The HTML element for the form.</td>
</tr>
</tbody>
</table>

**GlideForm - getHelpTextControl(String fieldName)**

Returns the HTML element of the help text for the specified field.

This method is applicable to service catalog variables only.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTMLElement</td>
<td>Help text field's HTML element.</td>
</tr>
</tbody>
</table>

**GlideForm - getIntValue(String fieldName)**

Returns the integer value of the field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Integer value of the field.</td>
</tr>
</tbody>
</table>

### GlideForm - `getLabelOf(String fieldName)`

Returns the plain text value of the field label.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The label text.</td>
</tr>
</tbody>
</table>

```javascript
if (g_user.hasRole('itil')) {
    var oldLabel = g_form.getLabelOf('comments');
    g_form.setLabelOf('comments', oldLabel + ' (Customer visible)');
}
```

### GlideForm - `getOption(String fieldName, String choiceValue)`

Returns the option element for a selected box named `fieldName` where `choiceValue` matches the option value.

#### Note: This method does not work on read-only fields.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
<tr>
<td>choiceValue</td>
<td>String</td>
<td>Value of the option.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTMLElement</td>
<td>The HTMLElement for the option. Returns null if the field or option is not found.</td>
</tr>
</tbody>
</table>
The following example shows how to get the label for a choice list value.

```javascript
// Get the label for a choice list value
// fieldName is 'category'
function onChange(control, oldValue, newValue, isLoading) {
    var choiceValue = g_form.getValue('category');
    var choiceLabel = g_form.getOption('category', choiceValue).text;
}
```

**GlideForm - getReference(String fieldName, Function callBack)**

Returns the GlideRecord for a specified field.

If a callback function is present, this routine runs asynchronously. The browser (and script) processing continues normally until the server returns the reference value, at which time, the callback function is invoked. If a callback function is not present, this routine runs synchronously and processing halts (causing the browser to appear to hang) while waiting on a server response.

**Important:** It is strongly recommended that you use a callback function.

Callback function support for ServiceCatalogForm.getReference is available.

**Note:** Using this method requires a call to the server which requires additional time and may introduce latency to your page. Use this method with caution. For additional information, see [Client script design and processing](#).

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
<tr>
<td>callBack</td>
<td>Function</td>
<td>Name of the call back function.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>GlideRecord object for the specified field. If the specified reference cannot be found, it returns an initialized GlideRecord object where currentRow = -1 and rows.length = 0.</td>
</tr>
</tbody>
</table>

**Code example**

```javascript
function onChange(control, oldValue, newValue, isLoading) {
    g_form.getReference('caller_id', doAlert); // doAlert is our callback function
}
```
function doAlert(caller) { // reference is passed into callback as first arguments
  if (caller.vip === 'true')
    alert('Caller is a VIP!');
}

GlideForm - getSectionNames()

Returns all section names, whether visible or not.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array of strings</td>
<td>The section names.</td>
</tr>
</tbody>
</table>

GlideForm - getSections()

Returns an array of the form's sections.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array of HTML elements</td>
<td>The form's sections.</td>
</tr>
</tbody>
</table>

function onChange(control, oldValue, newValue, isLoading) {
  // this example was run on a form divided into sections (Change form)
  // and hid a section when the "state" field was changed
  var sections = g_form.getSections();
  if (newValue == '2') {
    g_form.setSectionDisplay(sections[1], false);
  } else {
    g_form.setSectionDisplay(sections[1], true);
  }
}
GlideForm - getTableName()

Returns the name of the table to which this record belongs.

On the server side, the table for the current record can be retrieved with current.sys_class_name or current.getTableName().

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Name of the table.</td>
</tr>
</tbody>
</table>

```javascript
function onLoad() {
    if (g_form.isNewRecord()) {
        var tableName = g_form.getTableName(); //Get the table name
    }
}
```

GlideForm - getUniqueValue()

Returns the sys_id of the record displayed in the form.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The record's sys_id.</td>
</tr>
</tbody>
</table>

```javascript
function onLoad() {
    var incSysid = g_form.getUniqueValue();
    alert(incSysid);
}
```

GlideForm - getValue(String fieldName)

Returns the value of the specified field.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the specified field.</td>
</tr>
</tbody>
</table>

```javascript
function onChange(control, oldValue, newValue, isLoading) {
    alert(g_form.getValue('short_description'));
}
```

**GlideForm - hideAllFieldMsgs()**

Hides all field messages.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideForm - hideAllFieldMsgs(String type)**

Hides all field messages of the specified type.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>The type of message, info or error.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideForm - hideErrorBox(String fieldName)**

Hides the error message placed by showErrorBox().
Whenever possible, use `hideFieldMsg()` rather than this method whenever possible.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The name of the field or control.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideForm - hideFieldMsg(String fieldName)**

Hides the last message placed by `showFieldMsg()`.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideForm - hideFieldMsg(String fieldName, Boolean clearAll)**

Hides the messages placed by `showFieldMsg()`.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
<tr>
<td>clearAll</td>
<td>Boolean</td>
<td>When true, all messages for the field are cleared. When false, only the last message is removed.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
GlideForm - hideRelatedList(String listTableName)

Hides the specified related list on the form.
This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listTableName</td>
<td>String</td>
<td>Name of the related list. Use the sys_id to hide a list through a relationship.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideForm - hideRelatedLists()

Hides all related lists on the form.
This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideForm - isLiveUpdating()

Returns true while a live update is being done on the record the form is showing.
This can be used in an onChange() client script to determine if a change to the record is because of a live update from another session. The client script can then decide what action to take, or not to take. This applies to systems using UI16 with live forms enabled.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if a live update is happening on the record displayed by the form.</td>
</tr>
</tbody>
</table>

**GlideForm - isMandatory(String fieldName)**

Returns true if the field is mandatory.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field is required, false otherwise.</td>
</tr>
</tbody>
</table>

**GlideForm - isNewRecord()**

Returns true if the record has never been saved.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the record has not been saved; otherwise false.</td>
</tr>
</tbody>
</table>

```javascript
function onLoad() {
    if(g_form.isNewRecord()){
        alert('New Record!');
    }
}
```
**GlideForm - isSectionVisible(String sectionName)**

Returns true if the section is visible.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true when the section is visible; otherwise, false is returned.</td>
</tr>
</tbody>
</table>

**GlideForm - refreshSlushbucket(String fieldName)**

You can update a list collector variable.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the slush bucket.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
g_form.refreshSlushbucket('bucket');
```

**GlideForm - removeOption(String fieldName, String choiceValue)**

Removes the specified option from the choice list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
<tr>
<td>choiceValue</td>
<td>String</td>
<td>The value stored in the database. This is not the label.</td>
</tr>
</tbody>
</table>
GlideForm - removeDecoration(String fieldName, String icon, String title)
Removes the icon from the specified field that matches the icon and title.

**Note:** This method is not supported by Service Catalog.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Field name.</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>Name of the icon to remove.</td>
</tr>
<tr>
<td>title</td>
<td>String</td>
<td>The icon's text title (name).</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
function onChange(control, oldValue, newValue, isLoading) {
  // if the caller_id field is not present, then we can't add an icon anywhere
  if (!g_form.hasField('caller_id'))
    return;

  if (!newValue)
    return;

  g_form.getReference('caller_id', function(ref) {
    g_form.removeDecoration('caller_id', 'icon-star', 'VIP');
    if (ref.getValue('vip') == 'true')
      g_form.addDecoration('caller_id', 'icon-star', 'VIP');
  });
}
```

GlideForm - removeDecoration(String fieldName, String icon, String title, String color)
Removes the icon from the specified field that matches the icon, title, and color.
Note: This method is not supported by Service Catalog.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Field name.</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>Name of the icon to remove.</td>
</tr>
<tr>
<td>title</td>
<td>String</td>
<td>The icon's text title (name).</td>
</tr>
<tr>
<td>color</td>
<td>String</td>
<td>A CSS color</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
G._form.removeDecoration('caller_id', 'icon-star', 'VIP', 'blue');
```

**GlideForm - save()**

Saves the record without navigating away (update and stay).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideForm - setDisabled(String fieldName, Boolean disable)**

Makes the specified field available or unavailable.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
<tr>
<td>disable</td>
<td>Boolean</td>
<td>When true disables the field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When false enables the field.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideForm - setDisplay(String fieldName, Boolean display)

Displays or hides a field.
This method cannot hide a mandatory field with no value. If the field is hidden, the space is used to display other items. Whenever possible, use a UI policy instead of this method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
<tr>
<td>display</td>
<td>Boolean</td>
<td>When true displays the field, when false hides the field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
  //If the page isn't loading
  if (!isLoading) {
    //If the new value isn't blank
    if (newValue != '') {
      g_form.setDisplay('priority', false);
    } else
      g_form.setDisplay('priority', true);
  }
}
```

GlideForm - setLabelOf(String fieldName, String label)

Sets the plain text value of the field label.

**Note:** This method is not supported by Service Catalog.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
<tr>
<td>label</td>
<td>String</td>
<td>The field text label.</td>
</tr>
</tbody>
</table>
if (g_user.hasRole('itil')) {
    var oldLabel = g_form.getLabelOf('comments');
    g_form.setLabelOf('comments', oldLabel + ' (Customer visible)');
}

GlideForm - setMandatory(String fieldName, Boolean mandatory)

Makes the specified field mandatory.
Whenever possible, use a UI policy rather than this method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
<tr>
<td>mandatory</td>
<td>Boolean</td>
<td>When true makes the field mandatory. When false makes the field optional.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideForm - setReadOnly(String fieldName, Boolean readOnly)

Makes the specified field read only or editable.
Whenever possible, use a UI policy instead of this method.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
<tr>
<td>readOnly</td>
<td>Boolean</td>
<td>When true makes the field read only. When false makes the field editable.</td>
</tr>
</tbody>
</table>

Note: The function name `setReadonly()` also works.

To make a mandatory field read-only, you must first remove the mandatory requirement for that field by using the `setMandatory()` method.
### GlideForm - setValue(String fieldName, String value)

Sets the value of a field.

When defining a value in a choice list, be sure to use number value rather than the label.

To improve performance by preventing a round trip, include a display value in addition to the value, use `setValue(fieldName, value, displayValue)

**Note:** The method `setValue()` can cause a stack overflow when used in an OnChange client script. This is because every time the value is set, it will register as a change, which may re-trigger the OnChange client script. To prevent this, perform a check that will validate that the new value will be different from the old value. For example, before performing `setValue(shortDesc, newValue.toUpperCase());`, validate that the short description is not already uppercase. This will prevent the client script from applying the `toUpperCase()` more than once.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>Name of the field.</td>
</tr>
<tr>
<td>value</td>
<td>Value in the database.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
G_form.setValue('short_description', 'replace this with appropriate text');
```

### GlideForm - setValue(String fieldName, String value, String displayValue)

Sets the value and display value of the specified field.

When defining a value in a choice list, be sure to use number value rather than the label.

To improve performance by preventing a round trip when setting the value for a reference field, use this method not `setValue(fieldName, value)`. When setting multiple reference values for a list collector field, arrays can be passed in the second and third parameters.

**Note:** The method `setValue()` can cause a stack overflow when used in an OnChange client script. This is because every time the value is set, it will register as a change, which may re-trigger the OnChange client script. To prevent this, perform a check that will validate that the new value will be different from the old value. For example, before...
performing `setValue(shortDesc, newValue.toUpperCase());` validate that the short description is not already uppercase. This will prevent the client script from applying the `toUpperCase()` more than once.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>System ID for the reference value in the database. Can be an array of system IDs if the field is a glide-list.</td>
</tr>
<tr>
<td>displayValue</td>
<td>String</td>
<td>Display name for the referenced value in the database. Can be an array of display names if the field is a glide-list.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
g_form.setValue('assigned_to', userSysID, userName);
g_form.setValue('glide-list_field_name', sysIDArray, displayNameArray);
```

**GlideForm - setSectionDisplay(String sectionName, Boolean display)**

Shows or hides a section.
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sectionName</td>
<td>String</td>
<td>The section name is lower case with an underscore replacing the first space in the name, and with the remaining spaces being removed, for example &quot;Section Four is Here&quot; becomes &quot;section_fourishere&quot;. Other non-alphanumeric characters, such as ampersand (&amp;), are removed. Section names can be found by using the getSectionNames() method.</td>
</tr>
<tr>
<td>display</td>
<td>Boolean</td>
<td>When true shows the section. When false hides the section.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true when successful.</td>
</tr>
</tbody>
</table>

## GlideForm - setVisible(String fieldName, Boolean display)

Displays or hides the field.

On desktop UI, the space is left blank when hidden. On Mobile or Service Portal UI, the space is filled in my other fields when hidden. This method cannot hide mandatory fields with no value. Use UI Policy rather than this method whenever possible.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
<tr>
<td>display</td>
<td>Boolean</td>
<td>When true displays the field. When false hides the field.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
if(newValue != '') {
    g_form.setVisible('priority', false);
} else
    g_form.setVisible('priority', true);
}

GlideForm - showErrorBox(String name, String message, Boolean scrollForm)

Displays an error message under the specified form field (either a control object or the name of
the field). If the control or field is currently off the screen and the scrollForm parameter is true, the
form scrolls to the control or field.

A global property (glide.ui.scroll_to_message_field) is available that controls automatic
message scrolling when the form field is off screen (scrolls the form to the control or field). The
showFieldMsg() method is a similar method that requires a type parameter.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the field or control.</td>
</tr>
<tr>
<td>message</td>
<td>String</td>
<td>Message to display.</td>
</tr>
<tr>
<td>scrollForm</td>
<td>Boolean</td>
<td>When true scrolls the form to the field. When false the form does not scroll to the field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideForm - showFieldMsg(String field, String message, String type)

Displays either an informational or error message under the specified form field (either a control
object or the name of the field). If the control or field is off the screen, the form is scrolled to the
field.

A global property (glide.ui.scroll_to_message_field) is available that controls automatic message
scrolling when the form field is off screen (scrolls the form to the control or field).

The showErrorBox() method is a shorthand method that does not require the type parameter.

Note: This method does not work with the journal_field type field in UI16.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>String</td>
<td>Name of the field or control.</td>
</tr>
<tr>
<td>message</td>
<td>String</td>
<td>Message to display.</td>
</tr>
</tbody>
</table>
### GlideForm - `showFieldMsg(String field, String message, String type, Boolean scrollForm)`

Displays either an informational or error message under the specified form field (either a control object or the name of the field). If the control or field is currently off the screen and `scrollForm` is true, the form is scrolled to the field.

A global property (`glide.ui.scroll_to_message_field`) is available that controls automatic message scrolling when the form field is off screen (scrolls the form to the control or field).

The `showErrorBox()` method is a shorthand method that does not require the type parameter.

**Note:** This method does not work with the `journal_field` type field in UI16.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>String</td>
<td>Name of the field or control.</td>
</tr>
<tr>
<td>message</td>
<td>String</td>
<td>Message to display.</td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>&quot;error&quot;, &quot;info&quot;, or &quot;warning&quot;.</td>
</tr>
<tr>
<td>scrollForm</td>
<td>Boolean</td>
<td>When true, the form scrolls to the field if it is off screen. When false, the form does not scroll.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
// Example usage
g_form.showFieldMsg('impact','Low impact response time can be one week','info');
```

```javascript
// Example usage
g_form.showFieldMsg('impact','Low impact not allowed with High priority','error',false);
```
**GlideForm - showErrorBox(String name, String message)**

Displays an error message under the specified form field (either a control object or the name of the field). If the control or field is currently off the screen, the form scrolls to the control or field.

A global property (glide.ui.scroll_to_message_field) is available that controls automatic message scrolling when the form field is off screen (scrolls the form to the control or field). The showFieldMsg() method is a similar method that requires a type parameter.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the control or field.</td>
</tr>
<tr>
<td>message</td>
<td>String</td>
<td>The message to be displayed.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideForm - showRelatedList(String listTableName)**

Displays the specified related list on the form.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listTableName</td>
<td>String</td>
<td>Name of the related list.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideForm - showRelatedLists()**

Displays all the form's related lists.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
GlideForm - submit()
Saves the record.
The user is taken away from the form, returning them to where they were.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideForm - submit(String verb)
Performs the UI action specified by the parameter.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>verb</td>
<td>String</td>
<td>An action_name from a sys_ui_action record. The action name must be for a visible form button.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Mobile GlideForm (g_form)
Mobile GlideForm (g_form) methods enable you to work with forms on the mobile platform. Use these methods in any script that targets a mobile device.

MobileGlideForm (g_form) - addDecoration(String fieldName, String icon, String text)
Adds a decorative icon next to a field.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>The font icon to show next to the field.</td>
</tr>
<tr>
<td>text</td>
<td>String</td>
<td>The text title for the icon (used for screen readers).</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

This example adds a VIP icon next to the caller.

```javascript
function onChange(control, oldValue, newValue, isLoading) {
    // if the caller_id field is not present, then we can't add an icon anywhere
    if (!g_form.hasField('caller_id'))
        return;

    if (!newValue)
        return;

    g_form.getReference('caller_id', function(ref) {
        g_form.removeDecoration('caller_id', 'icon-star', 'VIP');
        if (ref.getValue('vip') == 'true')
            g_form.addDecoration('caller_id', 'icon-star', 'VIP');
    });
}
```

**MobileGlideForm (g_form) - getLabel(String fieldName)**

Gets the form label text.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The label text.</td>
</tr>
</tbody>
</table>
```javascript
if (g_user.hasRole('itil')) {
    var oldLabel = g_form.getLabel('comments');
    g_form.setLabel('comments', oldLabel + ' (Customer visible)');
}
```

**MobileGlideForm (g_form) - hasField(String fieldName)**

Determines if a field is present on the form.

Present means that it can be shown, not that it is visible.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field to look for.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field is present on the form; false, if it is not. On the form means that the field is part of g_form. It could still be hidden, read-only, mandatory, or invalid.</td>
</tr>
</tbody>
</table>

This example makes the assigned_to field mandatory if the assignment_group field is on the form.

```javascript
if (g_form.hasField('assignment_group'))
    g_form.setMandatory('assigned_to', true);
```

**MobileGlideForm (g_form) - removeDecoration(String fieldName, String icon, String text)**

Removes a decorative icon from next to a field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>The icon to remove.</td>
</tr>
<tr>
<td>text</td>
<td>String</td>
<td>The text title for the icon.</td>
</tr>
</tbody>
</table>
function onChange(control, oldValue, newValue, isLoading) {
    // if the caller_id field is not present, then we can't add an icon anywhere
    if (!g_form.hasField('caller_id')) {
        return;
    }
    if (!newValue) {
        return;
    }
    g_form.getReference('caller_id', function(ref) {
        g_form.removeDecoration('caller_id', 'icon-star', 'VIP');
        if (ref.getValue('vip') == 'true') {
            g_form.addDecoration('caller_id', 'icon-star', 'VIP');
        }
    });
}

MobileGlideForm (g_form) - setLabel(String fieldName, String label)
Sets the form label text.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
<tr>
<td>label</td>
<td>String</td>
<td>The field label text.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

This example changes the comments label.

```javascript
if (g_user.hasRole('itil')) {
    var oldLabel = g_form.getLabel('comments');
    g_form.setLabel('comments', oldLabel + ' (Customer visible)');
}
```
GlideFormScratchpad

The scoped GlideFormScratchpad class implements the `g_scratchpad` object for scoped applications.

The scoped GlideFormScratchpad class has no constructor and no methods. The `g_scratchpad` object behaves identically for global and scoped applications.

The `g_scratchpad` object provides a mechanism for passing information from the server to the client when the client requires information not available on a form. This can be accomplished by creating a business rule to put the information in the `g_scratchpad` object and accessing the information in a client script.

GlideGuidV3

You can create a globally unique identifier.

You access the GlideGuidV3 methods using the `g_guid` global object.

GlideGuidV3 - generate(Number stringLength)

Creates a globally unique identifier 32 characters long, or as specified with the optional length argument.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stringLength</td>
<td>Number</td>
<td>The desired string length, must be between 1 and 32 inclusive. This parameter is optional. If not specified, the returned string will be 32 characters long.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The globally unique identifier.</td>
</tr>
</tbody>
</table>

GlideImpersonate

The global GlideImpersonate API enables administrators to pose as another authenticated user for testing purposes.

Use the `GlideImpersonate` API when you need to act as another user so to test functionality within your application. When impersonating another user, the administrator has access to exactly what the impersonated user would have access to in the system, including the same menus and modules.

Note: The system records anything the administrator does while impersonating another user as having been done by that user.

GlideImpersonate - canDebug(String userSysId)

Verifies whether the specified user can perform debugging on scripts.
In order for a user to be able to debug scripts, they must be on a developer instance. Debugging is not allowed on production instances.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userSysId</td>
<td>String</td>
<td>sys_Id of the user to verify for debugging capability.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True: User is able to debug the application.</td>
</tr>
<tr>
<td></td>
<td>False: User is not able to debug the application.</td>
</tr>
</tbody>
</table>

```javascript
userDebug: function(userId) { 
    var impUser = new GlideImpersonate();
    impUser.canDebug(userId);
}
```

**GlideImpersonate - canImpersonate(String userSysId)**

Verifies whether the current user can impersonate the specified user.

If the current user is not assigned the admin role, the user to impersonate is inactive, or there are other issues with impersonating the specified user, the method returns “false” and the user cannot be impersonated.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userSysId</td>
<td>String</td>
<td>sys_Id of the user to impersonate</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True: Current user can impersonate the specified user.</td>
</tr>
<tr>
<td></td>
<td>False: Current user cannot impersonate the specified user.</td>
</tr>
</tbody>
</table>

```javascript
function onlineImpersonate(userSysId) {
    if (!GlideImpersonate().canImpersonate(userSysId)) {
        gs.addInfoMessage("No access to impersonate "+
        userSysId);
    } else {
```
GlideImpersonate - impersonate(String userSysId)

Sets the user ID for the current administrator to the passed-in user ID, enabling the administrator to act as that user.

When impersonating another user, the administrator has access to exactly what the impersonated user would have access to in the system, including the same menus and modules. Only use this method when testing functionality in an application. Ensure that once you are finished impersonating a user that you call the method again with the administrator sys_id to stop the impersonation.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userSysId</td>
<td>String</td>
<td>sys_id of the user to impersonate</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>sys_id of the user that was logged in prior to the impersonation request</td>
</tr>
</tbody>
</table>

GlideImpersonate - isImpersonating()

Determines whether the current user is impersonating another user.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True: User is currently impersonating another user. False: User is not currently impersonating another user.</td>
</tr>
</tbody>
</table>
```javascript
function abortOnImpersonate() {
    if (GlideImpersonate().isImpersonating()){
        current.setAbortAction(true);
        gs.addInfoMessage("Transaction canceled due to Impersonation");
    } return;
}
```

**GlideList2 (g_list)**

GlideList2 is a JavaScript class used to customize (v2) lists.

The variable `g_list` is used to access a specified list object. The `g_list` variable is not available to the related lists form link UI action. It is available to the lists form link UI action.

These methods are used in UI Context Menus and UI Actions.

**GlideList2 - addFilter(String filter)**

Adds a single term to the list query filter.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>String</td>
<td>Query string condition to add.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
g_list.addFilter("active=true");
```

**GlideList2 - get(Object DOMelement)**

Returns the GlideList2 object for the list that contains the specified item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOMelement</td>
<td>Object</td>
<td>The DOM element ID for the list for which you want the GlideList2 object.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The GlideList2 object or null if not found.</td>
</tr>
</tbody>
</table>

**GlideList2 - get(String ListID)**

Returns the GlideList2 object for the list specified.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListID</td>
<td>String</td>
<td>The list ID for which you want the GlideList2 object.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The GlideList2 object or null if not found.</td>
</tr>
</tbody>
</table>

```javascript
function assignLabelActionViaLookupModal(tableName, listId) {
  var list = GlideList2.get(listId);
  if (!list)
    return;
  else
    assignLabelViaLookup(tableName, sysIds, list.getView());
}
```

**GlideList2 - getChecked()**

Returns a comma-separated list of the sys_ids for the items that are checked in the list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Comma-separated list of the sys_ids for the items that are checked in the list. Does not check to determine that the items returned are allowed to be executed.</td>
</tr>
</tbody>
</table>
function removeLabelActionViaLookupModal(tableName, listId) {
    var list = GlideList2.get(listId);
    if (!list)
        return;

    var sysIds = list.getChecked();
    if (!sysIds)
        return;

    removeLabelViaLookup(tableName, sysIds);
}

GlideList2 - getFixedQuery()

Returns the sysparm_fixed query.
A fixed query is the part of the query that cannot be removed from the breadcrumb (i.e., it is fixed for the user). It is specified by including a sysparm_fixed_query parameter for the application module.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The fixed query string for the list.</td>
</tr>
</tbody>
</table>

var list = GlideList2.get(container.readAttribute('list_id'));
var filter = this._getFilter(element);
var fixedQuery = list.getFixedQuery();
if (fixedQuery)
    filter = fixedQuery + "^" + filter;

GlideList2 - getGroupBy()

Returns the field or comma-separated list of fields that are used to group the list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The field or comma-separated list of fields that are used to group the list.</td>
</tr>
</tbody>
</table>

```javascript
function runFilterV2Lists(name, filter) {
  var list = GlideList2.get(name);
  if (list) {
    var groupBy = list.getGroupBy();
    if (groupBy)
      filter += "^" + groupBy;
    list.setFilterAndRefresh(filter);
  }
}
```

**GlideList2 - getListName()**

Returns the name of the list, which is usually the table name.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The list name (usually the table name).</td>
</tr>
</tbody>
</table>

```javascript
var list = GlideList2.get(name);
var listName = list.getListName();
```

**GlideList2 - getOrderBy()**

Returns the first field used to order the list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The field used for order, or a blank.</td>
</tr>
</tbody>
</table>

```javascript
var list = GlideList2.get(listId);
if (!list)
    return;
var orderBy = list.getOrderBy();
```

#### GlideList2 - getParentTable()

Returns the name of the parent table for a related list (the table associated with the form).

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The parent table name.</td>
</tr>
</tbody>
</table>

```javascript
for (var id in GlideLists2) {
    var list = GlideLists2[id];
    if (list.getTableName() == listTableName &&
        list.getParentTable() == tableName)
        return list.getContainer();
}
```

#### GlideList2 - getQuery(Boolean orderBy, Boolean groupBy, Boolean fixed, Boolean all)

Returns the encoded query string for the list.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderBy</td>
<td>Boolean</td>
<td>(Optional) If true, includes the orderBy in the encoded query string.</td>
</tr>
<tr>
<td>groupBy</td>
<td>Boolean</td>
<td>(Optional) If true, includes the groupBy in the encoded query string.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>fixed</td>
<td>Boolean</td>
<td>(Optional) If true, includes fixed query in the encoded query string.</td>
</tr>
<tr>
<td>all</td>
<td>Boolean</td>
<td>(Optional) If true, includes orderBy, groupBy, and fixed query.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Encoded query string for the list.</td>
</tr>
</tbody>
</table>

```javascript
var list = GlideList2.get(this.listID);
var ajax = new GlideAjax("AJAXJellyRunner", "AJAXJellyRunner.do");
ajax.addParam("sysparm_query_encoded", list.getQuery({groupby: true, orderby: true}));
ajax.addParam("sysparm_table", list.getTableName());
ajax.addParam("sysparm_view", list.getView());
```

**GlideList2 - getRelated()**

Returns the related list field that associates the related list to the parent form.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Field that connects the list to the parent form.</td>
</tr>
</tbody>
</table>

```javascript
var list = GlideList2.get(name);
var related = list.getRelated();
if (related)
    ajax.addParam("sysparm_is_related_list", "true");
```

**GlideList2 - getTableName()**

Returns the table name for the list.
### GlideList2 - getListsForTable()

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Returns the table name for the list.</td>
</tr>
</tbody>
</table>

```javascript
GlideList2.getListsForTable = function(table) {
    var lists = [];
    for (var id in GlideLists2) {
        var list = GlideLists2[id];
        if (list.getTableName() == table)
            lists.push(list);
    }
    return lists;
}
```

### GlideList2 - getView()

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The name of the view.</td>
</tr>
</tbody>
</table>

```javascript
function assignLabelActionViaLookupModal(tableName, listId) {
    var list = GlideList2.get(listId);
    if (!list)
        return;
    assignLabelViaLookup(tableName, sysIds, list.getView());
}
```

### GlideList2 - getTitle()

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

Returns the list title.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The list title.</td>
</tr>
</tbody>
</table>

```javascript
var list = GlideList2.get(name);
var listTitle = list.getTitle();
```

**GlideList2 - isUserList()**

Returns **true** if the list has been personalized by the user by choosing the list mechanic and changing the list layout.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the list layout has been changed.</td>
</tr>
</tbody>
</table>

```javascript
var list = GlideList2.get(listId);
if (!list)
    return;
if (list.isUserList())
    var tableName = list.getTableName();
```

**GlideList2 - refresh(Number firstRow, String additionalParms)**

Refreshes the list. The **orderBy** part of the list filter is ignored so that the list uses its natural ordering when it is refreshed.

```javascript
var list = GlideList2.get(listId);
if (!list)
    return;
if (list.isUserList())
    var tableName = list.getTableName();
```
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstRow</td>
<td>Number</td>
<td>(Optional) The first row to appear in the list. If not specified, the first row of the current is used.</td>
</tr>
<tr>
<td>additionalParms</td>
<td>String</td>
<td>(Optional) name-value pairs that are submitted with the list refresh request.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
$timeout(function() {
  if (GlideList.lists) {
    var list = GlideList.get(name);
    if (list) {
      if (sortBy) {
        if (sortDirection == 'ASC')
          list.sort(sortBy);
        else
          list.sortDescending(sortBy);
      }
      list.refresh();
    }
  }
})
```

### GlideList2 - refreshWithOrderBy(NumberOf firstRow, String description)

Refreshes the list. The orderBy part of the list filter is included if it is specified for the list.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstRow</td>
<td>Number</td>
<td>(Optional) The first row to appear in the list.</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>(Optional) name=value pairs that are submitted with the list refresh request.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
ga.getXML(function(serverResponse) {
  var response = serverResponse.responseXML.getElementsByTagName("response")[0];
  if (response) {
    var list = GlideList2.getByName("backlog_stories");
    list.refreshWithOrderBy();
    var status = response.getAttribute('status');
    $j('html, body').animate({scrollTop:
      $j('#' + data.record.sys_id).offset().top}, 500);
    if (status == 'failure') {
      alert('${gs.getMessage("Story cannot be created. Team is not associated with any project.")}')
    }
  }
})
```

GlideList2 - setFilter(String filter)

Sets the encoded query string for the list, ignoring the orderBy and groupBy parts of the query string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>String</td>
<td>Encoded query string.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
list = GlideList2.get($(side + "ContentDivRelease").select(".list_div")
[0].getAttribute("id");
if (list) {
  list.setFilter("active=true");
  list.refresh(1);
}
```

GlideList2 - setFilterAndRefresh(String filter)

Sets the encoded query string for the list, including the orderBy and groupBy if specified, and then refreshes the list using the new filter.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>String</td>
<td>Encoded query string.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
function updateListFilter(projectID) {
  var list = GlideList2.getByName("backlog_stories");
  var fixedQuery = $('hdn_additional_filters').value;
  if(!projectID) {
    list.setFilterAndRefresh(fixedQuery + "^ORDERBYteam_index");
    list.setOrderBy("team_index");
  }
}
```

**GlideList2 - setFirstRow(Number rowNum)**

Sets the first row that appears in the list when the list is refreshed.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rowNum</td>
<td>Number</td>
<td>Row number of the first row to display.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var nextRow = 0;
var rowsPerPage = 20;
var list = GlideList2.get(listId);
if (!list) {
  return;
} else {
  list.setFirstRow(nextRow);
  nextRow = nextRow + rowsPerPage;
}
```

**GlideList2 - setGroupBy(String groupBy)**

Sets the `groupBy` criteria for the list, for a single field or multiple fields.
For a single field, use field or groupByField. The groupBy prefix is optional. For multiple fields, use field1^field2^field3 or groupByField1^groupByField2^groupByField3.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupBy</td>
<td>String</td>
<td>The groupBy criteria for the list.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
function runContextAction(listId) {
    var g_list = GlideList2.get(listId);
    g_list.setGroupBy('');
    g_list.refresh(1);
}
```

### GlideList2 - setOrderBy(String orderBy)

Sets the orderBy criteria for the list.

For a single order by field use orderBy field or orderByDescField. For multiple fields, use orderByField1^orderByField2^orderByField3. orderBy specifies ascending order and orderByDesc specifies descending. These prefix strings are optional. If not specified orderBy is assumed.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderBy</td>
<td>String</td>
<td>Single or multiple order by fields.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
updateOrderBy: function(orderBy){
    var list = GlideList2.get(this.listID);
    if (list)
        list.setOrderBy(orderBy);
};
```
GlideList2 - setRowsPerPage(Number rows)

Sets the number of rows per page to display.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rows</td>
<td>Number</td>
<td>The number of rows to display</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
link: function(scope) {
    var list = GlideList2.get(scope.listId);
    list.setRowsPerPage(scope.maxRows);
    list.setFilterAndRefresh(scope.tableQuery);
}
```

GlideList2 - showHideGroups(Boolean showFlag)

 Shows or hides all the groups within the list and saves the current collapsed/expanded state of the groups as a user preference.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showFlag</td>
<td>Boolean</td>
<td>If true, shows the groups within the list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
function showHideAllGroups(showFlag) {
    var list = GlideList2.get(listId);
    if (!list)
        return;
    list.showHideGroups(showFlag);
}
```
**GlideList2 - showHideList(Boolean showFlag)**
Displays or hides the list and saves the current collapsed/expanded state of the list as a user preference.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showFlag</td>
<td>Boolean</td>
<td>If true, displays the list.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
GlideList2.toggleAll = function(expandFlag) {
    for (var id in GlideLists2) {
        var list = GlideLists2[id];
        list.showHideList(expandFlag);
    }
}
```

**GlideList2 - sort(String field)**
Sorts the list in ascending order and saves the choice.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>String</td>
<td>Specifies the field used to sort the list.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
$timeout(function() {
    if (GlideList.lists) {
        var list = GlideList.get(name);
        if (list) {
            if (sortBy) {
                if (sortDirection == 'ASC')
                    list.sort(sortBy);
                else
                    list.sortDescending(sortBy);
            }
        }
    }
}
```
GlideList2 - sortDescending(String field, Number amount)

Sorts the list in descending order and saves the choice.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>String</td>
<td>Specifies the field used to sort the list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
function() {
  if (GlideList.lists) {
    var list = GlideList.get(name);
    if (list) {
      if (sortBy) {
        if (sortDirection == 'ASC')
          list.sort(sortBy);
        else
          list.sortDescending(sortBy);
      }
      list.refresh();
    }
  }
}
```

GlideList2 - toggleList()

Toggles the display of the list and saves the current collapsed/expanded state of the list as a user preference.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
var list = GlideList2.get(listId);
if (!list)
    return;
list.toggleList();

GlideList2 - toggleListNoPref()

Toggles the display of the list but does not save the current collapsed/expanded state of the list
as a user preference.

Parameters

Name | Type | Description
---- | ---- | -----------
none | none | None

Returns

Type | Description
---- | -----------
void | None

var list = GlideList2.get(listId);
if (!list)
    return;
list.toggleListNoPref();

GlideListV3 (g_list)

Use GlideListV3 to manipulate lists.

You access the GlideListV3 methods by using the g_list global object. These methods are used in UI
text context menus and UI actions. The g_list object is not available for related lists on the form link UI
action.

GlideListV3 - addFilter(String filter)

Adds a single term to the list query filter.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>String</td>
<td>Query string condition to add.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideListV3 - get(String listId)

Returns the GlideList object for specified list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listId</td>
<td>String</td>
<td>The list name.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The GlideList object for the specified list, or null if not found.</td>
</tr>
</tbody>
</table>

GlideListV3 - get(Object DomElement)

Returns the GlideList object for the specified DOM element.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomElement</td>
<td>Object</td>
<td>The DOM element ID for which you want the GlideList object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The GlideList object for the specified DOM element. Returns null if the DOM element is not found.</td>
</tr>
</tbody>
</table>

GlideListV3 - getChecked()

Returns a comma-separated list of sys_ids for checked items in the list. Does not return items that are not allowed to be executed.
### GlideListV3 - getFixedQuery()

Returns the sysparm_fixed query.

The fixed query is the part of the query that cannot be removed from the breadcrumb (i.e., it is fixed for the user). It is specified by including a `sysparm_fixed_query` parameter for the application module.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Comma-separated list of the sys_ids for checked items in the list. Does not return items that are not allowed to be executed.</td>
</tr>
</tbody>
</table>

### GlideListV3 - getFormTarget()

Returns the form's target attribute.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The fixed query string for the list.</td>
</tr>
</tbody>
</table>

### GlideListV3 - getGroupBy()

Returns the field or comma-separated list of fields that are used to group the list.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The form's target attribute.</td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The field or comma-separated list of fields used to group the list.</td>
</tr>
</tbody>
</table>

**GlideListV3 - getListName()**

Returns the name of the list, which is usually the table name.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The list name.</td>
</tr>
</tbody>
</table>

**GlideListV3 - getOrderBy()**

Returns the first field used to order the list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The field used to order the list, or an empty string if the list is not sorted.</td>
</tr>
</tbody>
</table>

**GlideListV3 - getParentTable()**

Returns the name of the parent table (the table associated with the form).
### GlideListV3 - `getQuery(Object options)`

Returns the encoded query string for the list.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| options | Object | The options can be one or more of the following.  
· orderby - include ORDERBY in the query  
· groupby - include GROUPBY in the query  
· fixed - include sysparm_fixed_query in the query  
· all - include all the options in the query |

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Encoded query string for the list.</td>
</tr>
</tbody>
</table>

### GlideListV3 - `getReferringUrl()`

Returns the referring URL.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.  
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
## GlideListV3 - getRelated()

Returns the related list field that associates the related list to the parent form.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Field that connects the list to the parent form.</td>
</tr>
</tbody>
</table>

## GlideListV3 - getRelatedListType()

Returns the related list type.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The relationship table type.</td>
</tr>
</tbody>
</table>

## GlideListV3 - getRelationshipId()

Returns the relationship record id, if this is type REL related list.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the relationship record.</td>
</tr>
</tbody>
</table>
**GlideListV3 - getRowCount()**

Retrieves the number of rows returned by the query.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of rows returned by the query.</td>
</tr>
</tbody>
</table>

**GlideListV3 - getRowsPerPage()**

Returns the number of rows to be displayed on a page.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of rows to be displayed on a page.</td>
</tr>
</tbody>
</table>

**GlideListV3 - getTableName()**

Returns the table name of the list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The list's table name.</td>
</tr>
</tbody>
</table>

**GlideListV3 - getTitle()**

Returns the list title.
GlideListV3 - getView()
Returns the view used to display the list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The list title.</td>
</tr>
</tbody>
</table>

GlideListV3 - isUserList()
Returns true if the list has been personalized by the user.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the list layout has changed.</td>
</tr>
</tbody>
</table>

GlideListV3 - refresh(Number firstRow, Object additionalParams)
Refreshes the list. The orderBy part of the list filter is ignored so that the list's natural ordering is used.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstRow</td>
<td>Number</td>
<td>(Optional) The first row to display in the list. If not specified, the list's current first row is used.</td>
</tr>
<tr>
<td>additionalParams</td>
<td>Object</td>
<td>(Optional) Name- value pairs that are submitted with the list refresh request.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### GlideListV3 - refreshWithOrderBy(Number firstRow, Object additionalParams)

Refreshes the list using the orderBy fields.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstRow</td>
<td>Number</td>
<td>(Optional) The first row to display in the list. If not specified, the list's current first row is used.</td>
</tr>
<tr>
<td>additionalParams</td>
<td>Object</td>
<td>(Optional) Name- value pairs that are submitted with the list refresh request.</td>
</tr>
</tbody>
</table>

### GlideListV3 - setFilter(String filter, Boolean saveOrderBy, Boolean saveGroupBy)

Sets the encoded query string for the list ignoring the orderBy and groupBy parts of the query string.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>String</td>
<td>An encoded query string.</td>
</tr>
<tr>
<td>saveOrderBy</td>
<td>Boolean</td>
<td>The default is false. When true uses the orderBy part of the query.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>saveGroupBy</td>
<td>Boolean</td>
<td>The default is false. When true uses the groupBy part of the query.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideListV3 - setFilterAndRefresh(String filter)**

Sets the encoded query string for the list, and then refreshes the list using the new filter. This preserves the groupby and orderby parameters.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>String</td>
<td>Encoded query string.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideListV3 - setFirstRow(Number firstRow)**

Sets the first row to be displayed when the list is refreshed.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstRow</td>
<td>Number</td>
<td>The row number in the list.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideListV3 - setFormTarget(String target)**

Specifies where to display the response from the form.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>String</td>
<td>The form.target attribute value to use.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### GlideListV3 - setGroupBy(String groupBy)

Sets the `groupBy` criteria for the list, for a single field or multiple fields.

For a single field, use field or `groupByField`. The `groupBy` prefix is optional. For multiple fields use `field1^field2^field3` or `groupByField1^groupByField2^groupByField3`.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>groupBy</td>
<td>The group by criteria for the list.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### GlideListV3 - setOrderBy(String orderBy)

Sets the `orderBy` criteria for the list.

For a single order by field use `orderBy` field or `orderByDescField`. For multiple fields, use `orderByField1^orderByField2^orderByField3`. `orderBy` specifies ascending order and `orderByDesc` specifies descending. These prefix strings are optional. If not specified `orderBy` is assumed.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderBy</td>
<td>String</td>
<td>Single or multiple order by fields.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
GlideListV3 - setReferringUrl(String url)
Sets the parent form referring url.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>String</td>
<td>The parent form's URL</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideListV3 - setRowsPerPage(Number numRows)
Set the number of rows to display on a page.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numRows</td>
<td>Number</td>
<td>The number of rows to display on a page.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideListV3 - showHideGroups(Boolean showFlag)
Displays or hides all of the groups within the list and saves the current collapsed/expanded state of the groups as a user preference.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showFlag</td>
<td>Boolean</td>
<td>When true, displays the groups within the list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
GlideListV3 - showHideList(Boolean showFlag)
Displays or hides the list and saves the current collapsed/expanded state of the list as a user preference.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showFlag</td>
<td>Boolean</td>
<td>When true, displays the list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideListV3 - sort(String field)
Sort the list in ascending order.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>String</td>
<td>The field to be used to sort the list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideListV3 - sortDescending(String field)
Sorts the list in descending order.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>String</td>
<td>The field used to sort the list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideListV3 - toggleList()
Toggles the list display between collapsed and expanded, and saves the state as a user preference.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideListV3 - toggleListNoPref()**

Toggles the list display between collapsed and expanded, but does not save the state as a user preference.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideListProperties**

Create a list and set list properties. For example, define whether a list has a filter, breadcrumbs, and search.

Use GlideListProperties in global server-side scripts. To use this class, instantiate a GlideListProperties object using the constructor.

For an example of this class in the base system, configure a list and select All. The tabbed list of options uses the personalize_all UI page and personalize_all_list UI macro to set list properties such as title, context menu, and breadcrumbs.

**GlideListProperties - GlideListProperties()**

Instantiates a GlideListProperties object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
```
**GlideListProperties - getListID()**

Returns the unique ID for a list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Unique ID for the list</td>
</tr>
</tbody>
</table>

```
var list = new GlideListProperties();
var getID = list.getListID();
gs.print(getID);
```

Output: 3519f77ad95f5700964f387107a8a394

**GlideListProperties - getListName()**

Returns the name of the list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Name of the list</td>
</tr>
</tbody>
</table>

```
var list = new GlideListProperties();
var setName = list.setListName("my custom list");
var getName = list.getListName();
gs.print(getName);
```

Output: my custom list

**GlideListProperties - getTitle()**

Get the title of a list.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The title of the list.</td>
</tr>
</tbody>
</table>

Optional example explanation

```javascript
var list = new GlideListProperties();
var title = list.setTitle("My title");
var getTitle = list.getTitle();
gs.print(getTitle);
```

Output: My title

GlideListProperties - hasActions()

Returns whether or not the **Actions on select rows** option is enabled for a list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns <strong>true</strong> if the actions option is enabled for a list.</td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var actions = list.setHasActions(true);
var hasActions = list.hasActions();
gs.print(hasActions);
```

Output: true

GlideListProperties - hasBottomNav()

Returns whether or not a list has navigation at the bottom.

© 2020 ServiceNow, Inc. All rights reserved.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If returns <code>true</code> the list has bottom navigation.</td>
</tr>
</tbody>
</table>

```java
var lp = new GlideListProperties();
var Nav = lp.setHasBottomNav(true);
var hasNav = lp.hasBottomNav();
gs.print(hasNav);
```

Output: true

**GlideListProperties - hasBottomVCR()**

Returns whether or not the page navigation controls appear in the footer of a list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If <code>true</code> the page navigation controls appear in the footer of a list.</td>
</tr>
</tbody>
</table>

```java
var list = new GlideListProperties();
var vcr = list.setHasBottomVCR(true);
var hasVCR = list.hasBottomVCR();
gs.print(hasVCR);
```

Output: true

**GlideListProperties - hasFilter()**

Returns whether or not a list has a filter.

The filter property is a parent of the breadcrumbs property. If the filter property is listed as false and the breadcrumb is listed as true, `hasFilter()` still returns true because the child property is marked as true.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If <strong>true</strong> a filter icon appears with the list, or the breadcrumb property is listed as true. If <strong>false</strong> both the filter property and the breadcrumb property are marked as false.</td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var filter = list.setHasFilter(true);
var breadcrumbs = list.setHasBreadcrumbs(true);
var hasFilter = list.hasFilter();
gs.print(hasFilter);
```

Output: true

**GlideListProperties - hasHeader()**

Returns whether or not a list has a header.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns <strong>true</strong> if a list has a header.</td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var header = list.setHasHeader(true);
var hasHeader = list.hasHeader();
gs.print(hasHeader);
```

Output: true

**GlideListProperties - hasHeaderContextMenu()**

Returns whether or not a header context menu is enabled for a list.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If <code>true</code> a context menu displays next to each column header in a list.</td>
</tr>
</tbody>
</table>

```java
var list = new GlideListProperties();
var header = list.setHeaderContextMenu(true);
var hasHeader = list.getHeaderContextMenu();
gs.print(hasHeader);
```

Output: `true`

### GlideListProperties - hasListMechanic()

Returns whether list personalization is enabled for a list.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If <code>true</code> the list mechanic is enabled for a list and the Personalize List icon appears on the page.</td>
</tr>
</tbody>
</table>

```java
var list = new GlideListProperties();
var mechanic = list.setHasListMechanic(true);
var hasMechanic = list.hasListMechanic();
gs.print(hasMechanic);
```

Output: `true`

### GlideListProperties - hasPopup()

Returns whether or not a list can have popup windows.
### GlideListProperties - hasPopup()

Returns true if the list allows popups.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var popup = list.setHasPopup(true);
var hasPopup = list.hasPopup();
gs.print(hasPopup);
```

Output: true

### GlideListProperties - hasRowContextMenu()

Returns whether or not rows in a list have a context menu.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var contextMenu = list.setHasRowContextMenu(true);
var hasContextMenu = list.hasRowContextMenu();
gs.print(hasContextMenu);
```

Output: true

### GlideListProperties - hasSearch()

Returns whether or not the search bar is enabled for a list.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var hasSearch = list.setHasSearch(true);
var hasSearch = list.hasSearch();
gs.print(hasSearch);
```
## GlideListProperties - hasSearch()

Returns whether or not the search bar appears in the header of a list.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If true the search bar appears in the header of a list.</td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var search = list.setHasSearch(true);
var hasSearch = list.hasSearch();
gs.print(hasSearch);
Output: true
```

## GlideListProperties - hasTitle()

Returns whether or not the list title appears in the list header.

The title context menu is a child property of title. If `setHasTitleContextMenu` is set to true, `hasTitle` also returns true, even if `setHasTitle` is set to false.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If true the list title appears in the list header.</td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var title = list.setHasTitle(true);
var contextMenu = list.setHasTitleContextMenu(true);
var hasTitle = list.hasTitle();
gs.print(hasTitle);
Output: true
```

## GlideListProperties - hasTitleContextMenu()

Returns whether a context menu appears in a list header.

The title context menu is a child property of title. If `setHasTitleContextMenu` is set to true, `hasTitle` also returns true, even if `setHasTitle` is set to false.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If true the context menu appears in the list header next to the list title.</td>
</tr>
</tbody>
</table>

Optional example explanation

```javascript
var list = new GlideListProperties();
var contextMenu = list.setHasTitleContextMenu(true);
var hasContextMenu = list.hasTitleContextMenu();
gs.print(hasContextMenu);
```

Output: true

**GlideListProperties - hasTopVCR()**

Returns whether or not the page navigation controls appear in the header of a list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If true the page navigation controls appear in the header of a list.</td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var vcr = list.setHasTopVCR(true);
var hasVCR = list.hasTopVCR();
gs.print(hasVCR);
```

Output: true

**GlideListProperties - isOmitFilter()**

Returns whether or not the omit filter option has been selected.

The ListControl omit flags take precedence in that if they are set, they negate the setting of their corresponding flag. For example, if the show filter flag has been set to true, but the ListControl omit filter is true, then checking hasFilter returns false.
### GlideListProperties - `isOmitFilter()`

Returns true or false whether the omit filter flag has been selected.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true or false whether the omit filter flag has been selected.</td>
</tr>
</tbody>
</table>

```javascript
var lp = new GlideListProperties();
var omitFilter = lp.isOmitFilter();
gs.print(omitFilter);
```

Output: false

### GlideListProperties - `isSaveFilterHidden()`

Returns whether the Save Filter button is hidden in the condition builder.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If true the Save Filter button is hidden in the condition builder.</td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var SaveFilter = list.setSaveFilterHidden(true);
var hasSaveFilter = list.isSaveFilterHidden();
gs.print(hasSaveFilter);
```

Output: true

### GlideListProperties - `isShowLinks()`

Returns whether or not a list shows links.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var list = new GlideListProperties();
var links = list.setShowLinks(true);
var hasLinks = list.isShowLinks();
gs.print(hasLinks);

Output: true

**GlideListProperties - isToggleHeader()**

Returns whether or not toggling the header columns is available for a list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If true, users can show or hide the column headers for a table.</td>
</tr>
</tbody>
</table>

var list = new GlideListProperties();
var toggle = list.setToggleHeader(true);
var hasToggle = list.isToggleHeader();
gs.print(hasToggle);

Output: true

**GlideListProperties - setCanChangeView(Boolean onOff)**

Determine whether the user can change the view for the list.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If false users cannot change the list view. By default, changing views is enabled.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var changeView = list.setCanChangeView(true);
```

**GlideListProperties - setCanGroup(Boolean onOff)**

Determine whether users can group items in a list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If false, the group by option does not appear in the column context menu. By default the group by option appears in the list context menu.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var canGroup = list.setCanGroup(true);
```

**GlideListProperties - setCanSort(Boolean onOff)**

Determine whether the sort option is available in a list.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If false, the sort option does not appear in column list context menu, and users cannot click the column title to change the order of the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var canSort = list.setCanSort(true);
```

### GlideListProperties - setContextMenus(Boolean onOff)

Displays or hides all of the available context menus for a list.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If set to true displays the title context menu, header context menu, and list context menu for a list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var context = list.setContextMenus(true);
```

### GlideListProperties - setHasActions(Boolean)

Determine whether the Actions on select rows options display at the bottom of a list.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true displays action options for a list.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```js
var list = new GlideListProperties();
var actions = list.setHasActions(true);
```

### GlideListProperties - setHasBottomNav(Boolean onOff)

Determine whether the navigation actions at the bottom of a list are hidden or not.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>When true adds navigation to the bottom of a list.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```js
var lp = new GlideListProperties();
var bottom = lp.setHasBottomNav(true);
```

### GlideListProperties - setHasBreadcrumbs(Boolean onOff)

Determine whether or not breadcrumbs appear at the top of a list. Breadcrumbs are a child of filters. To hide breadcrumbs completely, you need to also set the filter to false.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true breadcrumbs appear at the top of a list.</td>
</tr>
</tbody>
</table>
GlideListProperties - setHasBottomVCR(Boolean onOff)

Determine whether the first page, last page, next page, and previous page buttons appear at the bottom of the list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true, the first page, last page, next page, and previous page buttons appear at the bottom of the list.</td>
</tr>
</tbody>
</table>

GlideListProperties - setHasFilter(Boolean onOff)

Determine whether or not the filter displays as part of a list.

The filter is a parent of breadcrumbs. To remove the filter, you need to set both the filter and the breadcrumbs to false.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true a filter icon appears at the top of the list. Users can use the filter to narrow search results.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var filter = list.setHasFilter(true);
```

**GlideListProperties - setHasHeader(Boolean onOff)**

Determine whether or not a list displays a header.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true the list displays a header.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var header = list.setHasHeader(true);
```

**GlideListProperties - setHasHeaderContextMenu(Boolean onOff)**

Determine whether or not the context menu appears next to each column in a list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true the context menu appears next to each column in a list.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
```
var actions = list.setHeaderContextMenu(true);

**GlideListProperties - setHasListMechanic(Boolean onOff)**

Determine whether or not a list has the option for personalization.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If <code>true</code> the list mechanic is enabled and the Personalize List icon appears on the page.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var mechanic = list.setHasListMechanic(true);
```

**GlideListProperties - setHasPopup(Boolean onOff)**

Determine whether the list has a popup or modal window.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If <code>true</code> the list can have popup windows.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var popup = list.setHasPopup(true);
```

**GlideListProperties - setHasRowContextMenu(Boolean onOff)**

Determines whether or not list rows have a context menu.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>When true list rows can have a context menu.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var contextMenu = list.setHasRowContextMenu(true);
```

**GlideListProperties - setHasSearch(Boolean onOff)**

Determine whether search appears for a list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true the search bar appears in the list header.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var search = list.setHasSearch(true);
```

**GlideListProperties - setHasTitle(Boolean onOff)**

Determine whether the list title appears in the header.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true the title of the list appears in the list header.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var title = list.setTitle(true);
```

**GlideListProperties - setHasTitleContextMenu(Boolean onOff)**

Determine whether or not a list has a context menu in the header.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true the context menu appears next to the list title in the header.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var contextMenu = list.setTitleContextMenu(true);
```

**GlideListProperties - setHasTopVCR(Boolean onOff)**

Determine whether or not a list has the page navigation controls in the list header.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true the page navigation controls appear in the header of a list.</td>
</tr>
</tbody>
</table>
var list = new GlideListProperties();
var vcr = list.setHasTopVCR(true);

GlideListProperties - setHideRows(Boolean onOff)
Determine whether rows are visible in a list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true all of the rows are hidden for a list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var list = new GlideListProperties();
var hideRows = list.setHideRows(true);

GlideListProperties - setSaveFilterHidden(Boolean onOff)
Determine whether the Save Filter button appears in the condition builder.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true the Save Filter button is hidden.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var list = new GlideListProperties();
```javascript
var saveFilter = list.setSaveFilterHidden(true);

GlideListProperties - setListID(String sys_ID)
Set the unique ID for a list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique ID</td>
<td>String</td>
<td>The unique ID for the list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var setID = list.setListID("a9dd1483d99f5700964f387107a8a3ec");
var getID = list.getListID();
gs.print(getID);
```

Output: a9dd1483d99f5700964f387107a8a3ec

GlideListProperties - setListName(String name)
Defines a name for the list.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var name = list.setListName("my custom list");
```

GlideListProperties - setShowLinks(Boolean onOff)
Whether or not a list includes links.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true list includes related links.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var links = list.setShowLinks(true);
```

**GlideListProperties - setTitle(String title)**

Defines the list title.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td>Title for the list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var list = new GlideListProperties();
var title = list.setTitle("My title");
```

**GlideListProperties - setToggleHeader(Boolean onOff)**

Determine whether users can show or hide column headers for a table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If true an icon appears in the header that allows users to show or hide the column headers.</td>
</tr>
</tbody>
</table>
var list = new GlideListProperties();
var toggle = list.setToggleHeader(true);

**GlideListProperties - setVCR(Boolean onOff)**

Determine whether the first page, last page, next page, and previous page buttons appear at the top and bottom of the list.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOff</td>
<td>Boolean</td>
<td>If false, the list does not have any of the page navigation buttons for a list.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var list = new GlideListProperties();
var VCR = list.setVCR(true);

**GlideLocale**

GlideLocale provides information about display information for the local instance.

There is no constructor for a GlideLocale object. Use the get() method to get a GlideLocale object.

**Scoped GlideLocale - get()**

Returns the GlideLocale object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideLocale</td>
<td>The GlideLocale object.</td>
</tr>
</tbody>
</table>

```javascript
var locale = GlideLocale.get();
```

### Scoped GlideLocale - getDecimalSeparator()

Returns the decimal separator.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The decimal separator.</td>
</tr>
</tbody>
</table>

```javascript
var locale = GlideLocale.get();
var decimalSeparator = locale.getDecimalSeparator();
gs.info( "The decimal separator is " + decimalSeparator);
```

**Output:**

```
The decimal separator is .
```

### Scoped GlideLocale - getGroupingSeparator()

Returns the grouping separator.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var locale = GlideLocale.get();
```
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The grouping separator.</td>
</tr>
</tbody>
</table>

```javascript
var locale = GlideLocale.get();
var groupingSeparator = locale.getGroupingSeparator();
gs.info( "The grouping separator is " + groupingSeparator);
```

Output:

```
The grouping separator is ,
```

**GlideMenu (g_menu and g_item)**

GlideMenu methods are used in UI Context Menus, in the onShow scripts to customize UI Context Menu items.

There is no constructor for the GlideMenu class. Access GlideMenu methods using the g_menu global object.

- `g_menu` is the UI Context Menu that is about to be shown. The onShow script can make changes to the appearance of the menu before it is displayed using these methods.
- `g_item` is the current UI Context Menu item that is about to be shown. It is used in several of the g_menu methods to specify an item.

**GlideMenu - clearImage(GlideMenuItem item)**

Clears the image for an item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>GlideMenuItem</td>
<td>Specifies the item to have its image removed from display.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
g_menu.clearImage(g_item);
```

**GlideMenu - clearSelected()**

Clears any selection images from items in the menu.
### GlideMenu - getItem(String itemID)

Returns a menu item by item ID.

It can be necessary to find an item in a menu so that it can be changed before being displayed. Each menu item may be assigned a unique ID when the menu item is created (either from a UI Context Menu entry or from the `addAction()` method in the Dynamic Script Action).

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemID</td>
<td>String</td>
<td>Specifies the item to be returned.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideMenuItem</td>
<td>The menu item</td>
</tr>
</tbody>
</table>

### GlideMenu - setDisabled(GlideMenuItem item)

Disables a menu item so that it cannot be selected. The disabled menu item is displayed in a lighter color (grayed out) to indicate it is disabled.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>GlideMenuItem</td>
<td>The item to be disabled.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
**GlideMenu - setEnabled(GlideMenuItem item)**

Enables the specified menu item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>GlideMenuItem</td>
<td>The item to be enabled.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
g_menu.setEnabled(g_item);
```

**GlideMenu - setHidden(GlideMenuItem item)**

Hides the specified menu item.

When hiding menu items, the separator bars are not adjusted, so it is possible to end up with the menu showing two separators in a row.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>GlideMenuItem</td>
<td>The item to be hidden.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
g_menu.setHidden(g_item);
```

**GlideMenu - setImage(GlideMenuItem item, String imgSrc)**

Sets an image for an item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>GlideMenuItem</td>
<td>the item to have the image displayed.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
ServiceNow Kingston Now Platform Custom Business Applications

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imgSrc</td>
<td>String</td>
<td>the image to attach to the menu item.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
// Using GlideImages
g_menu.setImage(g_item, 'images/checked.gifx');
```

**GlideMenu - setLabel(GlideMenuItem item, String label)**

Sets the display label for a menu item. The label may contain HTML.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>GlideMenuItem</td>
<td>the item to be labeled.</td>
</tr>
<tr>
<td>label</td>
<td>String</td>
<td>the label to be displayed. The string may contain HTML.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
// Using GlideImages
g_menu.setLabel(g_item, "This is a new label");
```

**GlideMenu - setVisible(GlideMenuItem item)**

Displays the specified item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>GlideMenuItem</td>
<td>The item to be displayed.</td>
</tr>
</tbody>
</table>
GlideModalFormV3
Displays a form in a GlideModal.
General usage of the GlideModalForm class involves creating the object, setting any preferences, and then rendering the GlideModalForm.

```javascript
var d = new GlideModalForm('dialog title', 'table_name_or_form_name',
    [callback on completion of submit])
    d.setPreference('name', 'value');
    d.render();
```

Specify the query parameters that are passed to the form using `setPreference()`. Any name/value pair that you specify with `setPreference()` is sent along with the form POST request to display the form.

The GlideFormModal is set to fill the height of the document window.

GlideModalFormV3 - addParm(String name, String value)
Sets the specified parameter to the specified value.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The parameter name.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The parameter value.</td>
</tr>
</tbody>
</table>

GlideModalFormV3 - GlideModalForm(String title, String tableName, Function onCompletionCallback, Boolean readOnly)
Creates an instance of the GlideModalForm class.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td>The form title.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table being shown.</td>
</tr>
</tbody>
</table>
| onCompletionCallback  | Function| The function to call after the form has been submitted and processed on the server. The callback function has the form: `callbackFunction(String action_verb, String sys_id, String table, String displayValue)` where:  
  - `action_verb` is the name of the UI action executed. Examples are `sysverb_insert` (Submit button), `sysverb_cancel`, `sysverb_save` (Save button).  
  - `sys_id` is the sys_id of the affected record.  
  - `table` is the name of the table containing the record.  
  - `displayValue` |

```javascript
var d = new GlideModalForm('dialog title',  
  'table_name_or_form_name', [callback on completion of submit])  
  d.setPreference('name', 'value');  
  d.render();
```

**GlideModalFormV3 - render()**

Shows the form.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideModalFormV3 - setCompletionCallback(Function callbackFunction)**

Sets the function to be called when the form has been successfully submitted and processed by the server.
The callback function has the form `callbackFunction(String action_verb, String sys_id, String table, String displayValue)` where:

- `action_verb` is an `action_name` from a `sys_ui_action` record.
- `sys_id` is the `sys_id` of the affected record.
- `table` is the name of the table containing the record.
- `displayValue` is

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callbackFunction</td>
<td>Function</td>
<td>The callback function to be called when the form has been successfully processed.</td>
</tr>
</tbody>
</table>

### GlideModalFormV3 - setOnloadCallback(Function callbackFunction)

Sets the function to be called after the form has been loaded.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callbackFunction</td>
<td>Function</td>
<td>The function to be called after the form has been loaded. The callback function has the form <code>callBackFunction(GlideModalForm obj)</code></td>
</tr>
</tbody>
</table>

### GlideModalFormV3 - setSysID(String sys_id)

Sets the object's `sys_id` preference.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>The id preference. One of the query parameters passed to the form.</td>
</tr>
</tbody>
</table>
GlideModalV3

Provides methods for displaying a content overlay.

This is a fully-featured replacement for GlideWindow and GlideDialogWindow.

Example overlay

**GlideModalV3 - get(String id)**

Get a GlideModal object by ID.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>The element id of the GlideModal object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideModal</td>
<td>The object.</td>
</tr>
</tbody>
</table>

**GlideModalV3 - getPreference(String name)**

Returns the value of the specified property.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The property name</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The specified property's value</td>
</tr>
</tbody>
</table>

```javascript
var gm = new GlideModal('UI_dialog_name');
   //Sets the dialog title
gm.setTitle('Show title');
   //returns the value of the title
var title = gm.getPreference('title');
gm.setWidth(550);
   //Opens the dialog
gm.render();
```

**GlideModalV3 - GlideModal(String id, Boolean readOnly, Number width)**

Creates an instance of the GlideModalV3 class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>The UI page to load into the modal.</td>
</tr>
<tr>
<td>readOnly</td>
<td>Boolean</td>
<td>When true, hides the close button.</td>
</tr>
</tbody>
</table>
### GlideModalV3 - render()

Renders the UI page in the modal.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gm = new GlideModal("UI_dialog_name");
//Sets the dialog title
gm.setTitle('Show title');
gm.setWidth(550);
//Opens the dialog
gm.render();
```

### GlideModalV3 - renderWithContent(String html)

Display a modal with the specified HTML content.

The `renderWithContent()` method replaces the `render()` method, and does not request a UI page to render.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>html</td>
<td>String</td>
<td>The HTML content to be shown in the modal.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### GlideModalV3 - renderWithContent(Object html)

Display a modal with the specified HTML content.
The `renderWithContent()` method replaces the `render()` method, and does not request a UI page to render.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>html</td>
<td>Object</td>
<td>The HTML content to be shown in the modal.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideModalV3 - setPreference(String name, String value)**

Set a property that is read by the loaded UI page.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The property name</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The property value</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
var gm = new GlideModal('UI_dialog_name');
//Sets the dialog title
gm.setTitle('Show title');
gm.setPreference('table', 'task');
gm.setPreference('name', 'value');
//Opens the dialog
gm.render();
```

**GlideModalV3 - setPreferenceAndReload(Array properties)**

Set the properties and reload the modal.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>properties</td>
<td>Array</td>
<td>An array of name-value pairs to be set.</td>
</tr>
</tbody>
</table>
GlideModalV3 - setTitle(String title)
Sets the title of the modal.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td>The title to be displayed</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gm = new GlideModal('UI_dialog_name');
//Sets the dialog title
gm.setTitle('Show title');
gm.setPreference('name', 'value');
//Opens the dialog
gm.render();
```

GlideModalV3 - setWidth(Number width)
Set the width in pixels.
The modal is boxed into predefined system sizes.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>Number</td>
<td>The number of pixels.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gm = new GlideModal('UI_dialog_name');
//Sets the dialog title
gm.setTitle('Show title');
```
GlideModalV3 - switchView(String newView)
Change the view and reload the modal.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>newView</td>
<td>String</td>
<td>The view to use.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideNavigationV3
Provides methods to control and refresh the navigator and main frame.
The GlideNavigation methods are accessed using the g_navigation global object.

GlideNavigationV3 - open(String url, String target)
Redirects to a new URL.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>String</td>
<td>The URL to be loaded. It can be any URL supported by the browser.</td>
</tr>
<tr>
<td>target</td>
<td>String</td>
<td>The frame to use. If omitted, opens in the current frame.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideNavigationV3 - openPopup(String url, String name, String features, Boolean noStack)
Opens a popup window.
The features parameter is part of the DOM specification. The features are used and passed through. See the Mozilla Developer Network for more information on the feature list.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>String</td>
<td>The URL to open.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The window name</td>
</tr>
<tr>
<td>features</td>
<td>String</td>
<td>A comma separated list of features for the popup window.</td>
</tr>
<tr>
<td>noStack</td>
<td>Boolean</td>
<td>True to append sysparm_stack=no to the url. This prevents weirdness when using the form back button.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window</td>
<td>The instance of the new window.</td>
</tr>
</tbody>
</table>

**GlideNavigationV3 - openRecord(String tableName, String sys_id)**

Redirects to a record. The record will be displayed in the navigator.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The name of the table containing the record to be displayed.</td>
</tr>
<tr>
<td>sys_id</td>
<td>String</td>
<td>The sys_id of the record to be displayed.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideNavigationV3 - refreshNavigator()**

Refreshes the navigator display.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideNavigationV3 - reloadWindow()**

Reloads the current frame.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideNotificationV3**

You can show messages over the page content.

The GlideNotification method is accessed using the g_notification global object. List V3 must be activated for the g_notification object to be available.

**GlideNotificationV3 - show(String type, String message)**

Displays the specified string over the page content as the specified type of message.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>The type of message - error, warning, or info.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
// Displays an info message at the top of the screen
nowapi.g_notification.show("info", "The record has been updated");

// Displays an error message at the top of the screen
nowapi.g_notification.show("error", "You need to provide notes!");
```
GlideOAuthClient

Use these methods for requesting and revoking OAuth refresh and access tokens.

This API can be used in global and scoped scripts. In scoped scripts use the `sn_auth` namespace identifier.

GlideOAuthClient - getToken(String oauthEntityName, String requestor)

Retrieves the token for the client. You can use the token to check the expiration date and perform a token renewal.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OAuthEntityName</td>
<td>String</td>
<td>The OAuth entity.</td>
</tr>
<tr>
<td>requestor</td>
<td>String</td>
<td>The request.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScopedGlideOAuthToken</td>
<td>The token for the client.</td>
</tr>
</tbody>
</table>

This example code retrieves a token from the instance database:

```java
token = oAuthClient.getToken(testAppProvider, someone@someemail.com);
```

This example code retrieves a token using the password grant type.

```java
var tokenRequest = new sn_auth.GlideOAuthClientRequest();
tokenRequest.setGrantType("password");
tokenRequest.setUserName("itil");
tokenRequest.setPassword("itil");
tokenRequest.setRequestor("someone@someemail.com");
tokenRequest.setScope(null);

var oAuthClient = new sn_auth.GlideOAuthClient();
var tokenResponse = oAuthClient.requestTokenByRequest("testAppProvider", tokenRequest);
gs.info("Error:" + tokenResponse.getErrorMessage());

var token = tokenResponse.getToken();
dumpToken(token);
```

This example code retrieves a token from the instance database:

```java
token = oAuthClient.getToken(testAppProvider, someone@someemail.com);
dumpToken(token);
```

//call function below
This example code retrieves a token using the refresh token grant type.

```javascript
var tokenRequest = new sn_auth.GlideOAuthClientRequest();
tokenRequest.setGrantType("refresh_token");
tokenRequest.setRefreshToken(token.getRefreshToken());
tokenRequest.setRequestor("someone@someemail.com");

var oAuthClient = new sn_auth.GlideOAuthClient();
var tokenResponse = oAuthClient.requestTokenByRequest("testAppProvider",
tokenRequest);
gs.info("Error:" + tokenResponse.getErrorMessage());

var token = tokenResponse.getToken();
dumpToken(token);
//call function below
```

This function displays the access token, refresh token, and expiration date and time for the access token.

```javascript
function dumpToken(token) {
  if(token) {
    gs.info("AccessToken:" + token.getAccessToken());
    gs.info("AccessTokenExpiresIn:" + token.getExpiresIn());
    gs.info("RefreshToken:" + token.getRefreshToken());
  }
}
```

**GlideOAuthClient - requestToken(String clientName, String jsonString)**

Retrieves the token for the client, with the request parameters encoded in JSON format.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clientName</td>
<td>String</td>
<td>The client name.</td>
</tr>
<tr>
<td>jsonString</td>
<td>String</td>
<td>The JSON string for the client.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideOAuthClientResponse</td>
<td>The token for the client.</td>
</tr>
</tbody>
</table>

This example shows a resource owner password grant type request, with request parameters encoded in JSON format.

```javascript
var oAuthClient = new GlideOAuthClient();
var params = {grant_type:"password", username:"itil",
              password:'itil'};
var json = new JSON();
var text = json.encode(params);
```
var tokenResponse = oAuthClient.requestToken('TestClient', text);
var token = tokenResponse.getToken();

gs.log("AccessToken:" + token.getAccessToken());
gs.log("AccessTokenExpiresIn:" + token.getExpiresIn());
gs.log(" RefreshToken:" + token.getRefreshToken());

**GlideOAuthClient - requestTokenByRequest(String clientId, GlideOAuthClientRequest request)**

Retrieves the token for the client, with the client name and the request set into a GlideOAuthClientResponse object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clientId</td>
<td>String</td>
<td>The client name.</td>
</tr>
<tr>
<td>request</td>
<td>GlideOAuthClientRequest</td>
<td>The request.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideOAuthClientResponse</td>
<td>The token for the client.</td>
</tr>
</tbody>
</table>

**GlideOAuthClient - revokeToken(String clientId, String accessToken, String refreshToken, GlideOAuthClientRequest request)**

Revokes the access or refresh token for the client, with the request and optional header parameters set into a GlideOAuthClientRequest object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clientId</td>
<td>String</td>
<td>The client name.</td>
</tr>
<tr>
<td>accessToken</td>
<td>String</td>
<td>The access token.</td>
</tr>
<tr>
<td>refreshToken</td>
<td>String</td>
<td>The refresh token.</td>
</tr>
<tr>
<td>request</td>
<td>GlideOAuthClientRequest</td>
<td>The request.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideOAuthClientResponse</td>
<td>The token for the client.</td>
</tr>
</tbody>
</table>

**GlideOAuthClientRequest**

Use these methods for handling OAuth client requests.
This API can be used in global and scoped scripts. In scoped scripts use the `sn_auth` namespace identifier.

**GlideOAuthClientRequest - getGrantType()**

Retrieves the grant type.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The grant type.</td>
</tr>
</tbody>
</table>

**GlideOAuthClientRequest - getHeader(String name)**

Retrieves the HTTP headers for the string you provide.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the parameter.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringMap</td>
<td>The string map with the HTTP headers.</td>
</tr>
</tbody>
</table>

**GlideOAuthClientRequest - getHeaders()**

Retrieves the HTTP headers.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringMap</td>
<td>The string map with the HTTP headers.</td>
</tr>
</tbody>
</table>

**GlideOAuthClientRequest - getParameter(String name)**

Retrieves the parameters for the parameter name you provide.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The parameter name for which you want the parameters.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The parameters.</td>
</tr>
</tbody>
</table>

GlideOAuthClientRequest - getPassword()

Retrieves the password.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The password.</td>
</tr>
</tbody>
</table>

GlideOAuthClientRequest - getRefreshToken()

Retrieves the refresh token.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The refresh token.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The scope.</td>
</tr>
</tbody>
</table>

GlideOAuthClientRequest - `getUserName()`

Retrieves the user name.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The user name.</td>
</tr>
</tbody>
</table>

GlideOAuthClientRequest - `setGrantType(String grantType)`

Sets the grant type for the string you provide.

**Note:** You only need to set the grant type if it is not already defined in the OAuth provider profile.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The grant type.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideOAuthClientRequest - `setHead(String name, String value)`

Retrieves the HTTP headers for the string you provide.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the parameter.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value of the parameter.</td>
</tr>
</tbody>
</table>
### GlideOAuthClientRequest - setParameter(String name, String value)
Sets the parameters for the name:value pair of strings you provide.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The parameter name for which you want the parameters.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value of the parameter.</td>
</tr>
</tbody>
</table>

### GlideOAuthClientRequest - setPassword(String password)
Sets the password with the string you provide.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>String</td>
<td>The user name.</td>
</tr>
</tbody>
</table>

### GlideOAuthClientRequest - setRefreshToken(String refreshToken)
Sets the refresh token with the string you provide.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>refreshToken</td>
<td>String</td>
<td>The refresh token.</td>
</tr>
</tbody>
</table>
This example shows a resource owner password grant type request.

```javascript
var tokenRequest = new GlideOAuthClientRequest();
tokenRequest.setGrantType("password");
tokenRequest.setUserName("itil");
tokenRequest.setPassword("itil");
tokenRequest.setScope(null);

var oAuthClient = new GlideOAuthClient();
var tokenResponse = oAuthClient.requestToken("TestClient", tokenRequest);
gs.log("Error:" + tokenResponse.getErrorMessage());

var token = tokenResponse.getToken();
if (token) {
    gs.log("AccessToken:" + token.getAccessToken());
    gs.log("AccessTokenExpiresIn:" + token.getExpiresIn());
    gs.log("RefreshToken:" + token.getRefreshToken());
}
```

This example shows a refresh token grant type request.

```javascript
var tokenRequest = new GlideOAuthClientRequest();
tokenRequest.setGrantType("refresh_token");
tokenRequest.setRefreshToken("N-GtdSVLkWP_Cr-TysXdmNy59ZYafu5ZzAS4YaSluXDM0kCkInEnu-hwM5sGYSFwKJ6xaVmoaq7xJNoalXFQ");
tokenRequest.setScope(null);

var oAuthClient = new GlideOAuthClient();
tokenResponse = oAuthClient.requestToken("TestClient", tokenRequest);
gs.log("Error:" + tokenResponse.getErrorMessage());
token = tokenResponse.getToken();
if (token) {
    gs.log("AccessToken:" + token.getAccessToken());
    gs.log("AccessTokenExpiresIn:" + token.getExpiresIn());
    gs.log("AccessTokenSysID:" +
        token.getAccessTokenSysID());
    gs.log("RefreshToken:" + token.getRefreshToken());
    gs.log("RefreshTokenSysID:" +
        token.getRefreshTokenSysID());
}
```

**GlideOAuthClientRequest - setScope(String scope)**

Sets the scope for the string you provide.

**Note:** You only need to set the scope if it is not already defined in the OAuth provider.
**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope</td>
<td>String</td>
<td>The scope.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideOAuthClientRequest - setUserName(String userName)**

Sets the user name with the string you provide.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userName</td>
<td>String</td>
<td>The user name.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideOAuthClientResponse**

Use these methods for handling OAuth client responses.

This API can be used in global and scoped scripts. In scoped scripts use the `sn_auth` namespace identifier.

**GlideOAuthClientResponse - getBody()**

Retrieves all of the response information, including instance information.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The response information.</td>
</tr>
</tbody>
</table>

**GlideOAuthClientResponse - getContentType()**

Retrieves the HTTP response content header from an external OAuth provider.
### GlideOAuthClientResponse - `getErrorMessage()`

Retrieves the error message if authentication is not successful.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The error message.</td>
</tr>
</tbody>
</table>

### GlideOAuthClientResponse - `getResponseCode()`

Retrieves the HTTP response code from the external OAuth provider.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The HTTP response code.</td>
</tr>
</tbody>
</table>

### GlideOAuthClientResponse - `getResponseParameters()`

Retrieves the error message if authentication is not successful.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The HTTP response header.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MapString</td>
<td>The response content.</td>
</tr>
</tbody>
</table>

**GlideOAuthClientResponse - getToken()**

Retrieves the refresh token.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideOAuthToken</td>
<td>The refresh token.</td>
</tr>
</tbody>
</table>

**GlideOAuthToken**

Use the GlideOAuthToken methods for retrieving OAuth access token and information about the access token.

This API can be used in global and scoped scripts. In scoped scripts use the `sn_auth` namespace identifier.

**GlideOAuthToken - getAccessToken()**

Retrieves the access token.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The access token.</td>
</tr>
</tbody>
</table>

**GlideOAuthToken - getAccessTokenSysID()**

Retrieves the sys_id of the token ID.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the access token.</td>
</tr>
</tbody>
</table>

**GlideOAuthToken - getExpiresIn()**
Retrieves the lifespan of the access token in seconds.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The lifespan.</td>
</tr>
</tbody>
</table>

**GlideOAuthToken - getRefreshToken()**
Retrieves the lifespan of the access token in seconds.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The refresh token.</td>
</tr>
</tbody>
</table>

**GlideOAuthToken - getRefreshTokenSysID()**
Retrieves the sys_id of the refresh token.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The sys_id of the refresh token.</td>
</tr>
</tbody>
</table>
GlideOAuthToken - getScope()

Retrieves the scope, which is the amount of access granted by the access token.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The scope.</td>
</tr>
</tbody>
</table>

GlidePluginManager

The scoped GlidePluginManager API provides a method for determining if a plugin has been activated.

Scoped GlidePluginManager - isActive(String pluginID)

Determines if the specified plugin has been activated.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pluginID</td>
<td>String</td>
<td>The plugin ID</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the plugin has been activated.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('sys_plugins');
var queryString = "active=0^ORactive=1";
gr.addEncodedQuery(queryString);
gr.query();
pMgr = new GlidePluginManager();

while (gr.next()) {
   var name = gr.getValue('name');
   var pID = gr.getValue('source');
   isActive = pMgr.isActive(pID);
   if (isActive)
      gs.info('The plugin ' + name + " is active" );
}
```

Output:

The plugin Country Lookup Data is active
The plugin Database Replication is active
The plugin REST API Provider is active
The plugin Ten Cool Things is active
...

GlideQueryCondition

The GlideQueryCondition API provides additional AND or OR conditions that can be added to the current condition, allowing you to build complex queries.

Build complex queries such as:

```text
category='hardware' OR category='software' AND priority='2' AND priority='1'
```

In the case of addCondition(), an implied AND is added.

This class has no constructor. A GlideQueryCondition object is returned by the following methods:

- addActiveQuery()
- addInactiveQuery()
- addJoinQuery()
- addNotNullQuery()
- addNullQuery()
- addQuery()

If there is a complicated set of AND and OR queries, a single encoded query containing all conditions simplifies the query creation. To simplify the query creation, create a query in a list view, right-click the query, and select Copy query. It creates a single encoded query string to return your result set. Use that string as a parameter in an addEncodedQuery() call.

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an insert(), update(), deleteRecord(), or deleteMultiple() method on bad query results can result in data loss.

You can set the glide.invalid_query.returns_no_rows system property to true to have queries with invalid encoded queries return no records.

GlideQueryCondition - addCondition(String name, String oper, Object value)

Adds an AND condition to the current condition.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of a field.</td>
</tr>
<tr>
<td>oper</td>
<td>String</td>
<td>(Optional) The operator for the query. If you do not specify an operator, the condition uses an equals operator.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>The value to query on.</td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>A reference to a GlideQueryCondition that was added to the GlideRecord.</td>
</tr>
</tbody>
</table>

```javascript
class GlideRecord {
    constructor(name) {
        this.name = name;
    }
}

var gr = new GlideRecord('incident');
var qc = gr.addQuery('category', 'Hardware');
qc.addCondition('category', 'Network');
gr.addQuery('number', 'INC0000003');
gr.next();
gr.number;
gs.info(gr.getEncodedQuery());
```

### Scoped equivalent

To use the `addCondition()` method in a scoped application, use the corresponding scoped method: `addCondition()`.

### GlideQueryCondition - addOrCondition(String name, String oper, Object value)

Appends a 2-or-3 parameter OR condition to an existing GlideQueryCondition.

- `addOrCondition()` works in conjunction with any of the `addQuery()` methods to OR the specified query parameters to the query previously constructed using `addQuery()`.

- `addOrCondition()` is typically called with three parameters: table field, operator, and comparison value. It can be called with only two parameters, table field and comparison value, such as `qc.addOrCondition('category', 'software');`. The operator in this case is assumed to be “equal to”.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Field name</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| oper | String | (Optional) Query operator. The available values are dependent on the data type of the value parameter. Numbers:  
- =  
- !=  
- >  
- >=  
- <  
- <=  
Strings (must be in upper case):  
- =  
- !=  
- IN  
- STARTSWITH  
- ENDSWITH  
- CONTAINS  
- DOESNOTCONTAIN |
| value | Object | Value on which to query (not case-sensitive). |

**Note:** All passed in arrays must contain a minimum of two elements. Single element arrays are not supported.

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>A reference to a GlideQueryCondition that was added to the GlideRecord.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
var qc = gr.addQuery('category', 'Hardware');
qc.addOrCondition('category', 'Network');
gr.addQuery('number','INC0000003');
gr.next();
gr.number;
gs.info(gr.getEncodedQuery());
```
To group AND/OR statements such as \((\text{state} < 3 \text{ OR state} > 5) \text{ AND} \) \((\text{priority} = 1 \text{ OR priority} = 5)\) use code similar to the following:

```javascript
var myObj = new GlideRecord('incident');
var q1 = myObj.addQuery('state', '<', 3);
q1.addOrCondition('state', '>', 5);
var q2 = myObj.addQuery('priority', 1);
q1.addOrCondition('priority', 5);
myObj.query();
```

**Scoped equivalent**

To use the `addOrCondition()` method in a scoped application, use the corresponding scoped method: `addOrCondition()`.

**GlideQueryCondition**

The scoped GlideQueryCondition API provides additional AND or OR conditions that can be added to the current condition, allowing you to build complex queries.

Build complex queries such as:

```
category='hardware' OR category='software' AND priority='2' AND priority='1'
```

In the case of `addCondition()`, an implied AND is added.

This class has no constructor. A GlideQueryCondition object is returned by the following methods:

- `addActiveQuery()`
- `addInactiveQuery()`
- `addJoinQuery()`
- `addNotNullQuery()`
- `addNullQuery()`
- `addQuery()`

If there is a complicated set of AND and OR queries, a single encoded query simplifies the query creation. To simplify the query creation, create a query in a list view, right-click the query, and select **Copy query**. It creates a single encoded query string to return your result set. Use that string as a parameter in an `addEncodedQuery()` call.

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an `insert()`, `update()`, `deleteRecord()`, or `deleteMultiple()` method on bad query results can result in data loss.

You can set the glide.invalid_query.returns_no_rows system property to true to have queries with invalid encoded queries return no records.

**Scoped GlideQueryCondition - addCondition(String name, String oper, Object value)**

Advis an AND condition to the current condition.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of a field.</td>
</tr>
<tr>
<td>oper</td>
<td>String</td>
<td>(Optional) The operator for the query. If you do not specify an operator, the condition uses an equals operator.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>The value to query on.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>A reference to a GlideQueryCondition that was added to the GlideRecord.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
var qc = gr.addQuery('category', 'Hardware');
qc.addCondition('category', 'Network');
gr.addQuery('number', 'INC0000003');
gr.next();
gr.number;
gs.info(gr.getEncodedQuery());
```

Scoped GlideQueryCondition - addOrCondition(String name, String oper, Object value)

Appends a 2-or-3 parameter OR condition to an existing GlideQueryCondition.

addOrCondition() works in conjunction with any of the addQuery() methods to OR the specified query parameters to the query previously constructed using addQuery().

addOrCondition() is typically called with three parameters; table field, operator, and comparison value. It can be called with only two parameters, table field and comparison value, such as qc.addOrCondition('category', 'software');. The operator in this case is assumed to be "equal to".

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Field name</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>oper</td>
<td>String</td>
<td>(Optional) Query operator. The available values are dependent on the data type of the value parameter. Numbers: • = • != • &gt; • &gt;= • &lt; • &lt;= Strings (must be in upper case): • = • != • IN • STARTSWITH • ENDSWITH • CONTAINS • DOESNOTCONTAIN</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>Value on which to query (not case-sensitive).</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>A reference to a GlideQueryCondition that was added to the GlideRecord.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
var qc = gr.addQuery('category', 'Hardware');
qc.addOrCondition('category', 'Network');
gr.addQuery('number', 'INC0000003');
gr.next();
gr.number;
gs.info(gr.getEncodedQuery());
```
To group AND/OR statements to make complex queries, such as "All incidents with a (state less than 3 OR greater than 5) AND (priority is 1 OR priority is 5)"

```javascript
var myObj = new GlideRecord('incident');
var q1 = myObj.addQuery('state', '<', 3);
q1.addOrCondition('state', '>', 5);
var q2 = myObj.addQuery('priority', 1);
q2.addOrCondition('priority', 5);
myObj.query();
```

**GlideRecord**

GlideRecord is used for database operations. The client-side GlideRecord API enables the use of some GlideRecord functionality in client-side scripts, such as client scripts and UI policy scripts.

A GlideRecord contains both records and fields. Queries made with the client-side GlideRecord are executed on the server. Therefore, a request is made from the client browser to obtain the record data.

The client-side GlideRecord API is not supported in scoped applications. Instead, create a script include and use the GlideAjax API, or use the REST APIs. In addition, the client-side GlideRecord API applies ACLs based on the credentials of the user executing the script. To execute the code on the server without ACLs, use the GlideAjax API.

**GlideRecord - addOrderBy(String column)**

Adds a column to order by in the query.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>String</td>
<td>The column by which to order the result set.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideRecord - addQuery(String query)**

Adds a filter to return records using an encoded query string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>String</td>
<td>An encoded query string.</td>
</tr>
</tbody>
</table>
GlideRecord - addQuery(String name, Object value)

Adds a query to return records where the specified field name is equal to a specified value (or is in a list of values).

If you are familiar with SQL, this method is similar to the "where" clause. You can create one or more queries for a single filter by calling this method multiple times; for this method the queries are AND'ed. Once you define all of the desired queries, call the GlideRecord - query(Function responseFunction) to execute the specified query clause (filter).

To perform an operation other than AND, use the addQuery(String name, Object operator, Object value) method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the field to check.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>Value on which to query.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example showing multiple queries.

```javascript
var gr = new GlideRecord('incident');
gr.addQuery('priority', 4); // Priority is 4 - Low and,
gr.addQuery('state', 3); // State is On Hold
gr.query(response);

function response(result) {
  while(result.next()) {
    // Print all INC with priority 4 - Low AND state is On Hold
    console.log(result.getValue('number'));
  }
}
```

Example showing how to pass a string object.

```javascript
var gr = new GlideRecord('incident');
gr.addQuery('short_description', new String('USB device not working')); // string object
gr.addQuery('priority', 4); // number
gr.query(response);
```
function response(result) {
    while(result.next()) {
        // Print all INC with priority 4 - Low AND short
        description contains 'USB device not working'
        console.log(result.getValue('number'));
    }
}

**GlideRecord - addQuery(String name, Object operator, Object value)**

Adds a filter to return records where the field meets the specified condition (field, operator, value).

If you are familiar with SQL, this method is similar to the "where" clause. You can create one or more queries for a single filter by calling this method multiple times. Once you define all of the desired queries, call the **GlideRecord - query(Function responseFunction)** to execute the specified query clause (filter).

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the field to check.</td>
</tr>
</tbody>
</table>
| operator | Object  | Query operator. The available values are dependent on the data type of the value parameter. Numbers:  
|          |         | • =  
|          |         | • !=  
|          |         | • >  
|          |         | • >=  
|          |         | • <  
|          |         | • <=  
|          |         | Strings (must be in upper case):  
|          |         | • =  
|          |         | • !=  
|          |         | • IN  
|          |         | • NOT IN  
|          |         | • STARTSWITH  
|          |         | • ENDSWITH  
|          |         | • CONTAINS  
|          |         | • DOES NOT CONTAIN  
|          |         | • INSTANCEOF  
| value    | Object  | Value on which to query (not case-sensitive).    |
Example showing how to add multiple queries to a filter.

```javascript
var gr = new GlideRecord('incident');
gr.addQuery('priority', '<=', 2); // Priority is 2 or higher
and,
gr.addQuery('short_description', 'CONTAINS', 'crash'); // Short
description contains the word crash
gr.query(response);

function response(result) {
    while(result.next()) {
        // Print all INC with priority of 2 or higher AND short
description contains "crash"
        console.log(result.getValue('number'));
    }
}
```

Example showing how to pass in an array to verify multiple conditions in a single query.

```javascript
var priorities = [4, 2];
var gr = new GlideRecord('incident');
gr.addQuery('priority', 'IN', priorities);
gr.query(response);

function response(result) {
    while(result.next()) {
        console.log(result.getValue('number'));
    }
}
```

**GlideRecord - deleteRecord(Function responseFunction)**

Deletes the current record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>responseFunction</td>
<td>Function</td>
<td>The response function for the Ajax callback.</td>
</tr>
</tbody>
</table>
### GlideRecord - get(Object sys_id)

Executes a GlideRecord query for a record with the specified sys_id. This method is expected to be used to query for single records, so a next operation is performed before returning.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>Object</td>
<td>The sys_id of the record to be found.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if one or more matching records was found. False if no records were found.</td>
</tr>
</tbody>
</table>

### GlideRecord - getEncodedQuery()

Retrieves the query condition of the current result set as an encoded query string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The encoded query as a string.</td>
</tr>
</tbody>
</table>

### GlideRecord - getLimit()

Returns the limit for records to be returned by the GlideRecord query.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>The limit for records to be returned by the GlideRecord query.</td>
<td></td>
</tr>
</tbody>
</table>

**GlideRecord - getTableName()**

Retrieves the name of the table associated with this GlideRecord.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The table name</td>
</tr>
</tbody>
</table>

```javascript
var item = new GlideRecord('sc_request');
item.addQuery('sys_id', current.request);
item.query(itemResponse);

function itemResponse(item) {
    alert('The table is ' + item.getTableName());
}
```

**GlideRecord - GlideRecord(String tableName)**

Creates an instance of the GlideRecord class for the specified table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table to be used.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
```

**GlideRecord - hasNext()**

Determines if there are any more records in the GlideRecord.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if there are more records in the query set.</td>
</tr>
</tbody>
</table>

**GlideRecord - insert(Function responseFunction)**

Inserts a new record using the field values that have been set for the current record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>responseFunction</td>
<td>Function</td>
<td>The response function.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the inserted record, or null if the record was not inserted.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('to_do');
gr.initialize();
gr.name = 'first to do item';
gr.description = 'learn about GlideRecord';
gr.insert();
```

**GlideRecord - next()**

Moves to the next record in the GlideRecord.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>False if there are no more records in the query set.</td>
</tr>
</tbody>
</table>

```javascript
var rec = new GlideRecord('incident');
rec.query(recResponse);

function recResponse(rec) {
    while (rec.next()) {
        alert(rec.number + ' exists');
    }
}
```

### GlideRecord - orderBy(String column)

Specifies an orderBy column. May be called more than once to order by multiple columns.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>String</td>
<td>The column name to be used to order the result set.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
function UpdateProjectWBS(project) {
    var count = 0;
    var child = new GlideRecord('pm_project_task');
    child.addQuery('parent', project.sys_id);
    child.orderBy('order');
    child.orderBy('number');
    child.query(childResponse);
    g_form.addInfoMessage(count + ' Project Tasks updated');
}

function childResponse(child) {
    var len = child.getRowCount().toString().length;
    var seq = 0;
    while (child.next()) {
        count += UpdateProjectTaskWBS(child, 1, ++seq, len, '');
    }
}
```
**GlideRecord - query(Function responseFunction)**

Runs the query against the table based on the addQuery() filter. This queries the GlideRecord table as well as any references of the table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>responseFunction</td>
<td>Function</td>
<td>The response function for the Ajax callback.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var rec = new GlideRecord('incident');
rec.query(recResponse);

function recResponse(rec) {
  while (rec.next()) {
    alert(rec.number + ' exists');
  }
}
```

**GlideRecord - setLimit(Number maxQuery)**

Sets the limit for how many records are in the GlideRecord.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxQuery</td>
<td>Number</td>
<td>The limit for the number of records to retrieve.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideRecord**

GlideRecord is used for database operations.

A GlideRecord contains both records and fields. For information about GlideRecordSecure, which is a class inherited from GlideRecord that performs the same functions as GlideRecord, and also enforces ACLs, see the [GlideServer APIs](#).
Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an `insert()`, `update()`, `deleteRecord()`, or `deleteMultiple()` method on bad query results can result in data loss.

You can set the `glide.invalid_query.returns_no_rows` system property to true to have queries with invalid encoded queries return no records.

**GlideRecord - `addActiveQuery()`**

Adds a filter to return active records.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryCondition</td>
<td>Filter to return active records.</td>
</tr>
</tbody>
</table>

```javascript
var inc = new GlideRecord('incident');
inc.addActiveQuery();
inc.query();
```

**Scoped equivalent**

To use the `addActiveQuery()` method in a scoped application, use the corresponding scoped method: `addActiveQuery()`.

**GlideRecord - `addDomainQuery(Object glideRecord)`**

Changes the domain used for the query from the user’s domain to the domain of the provided `GlideRecord`.

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an `insert()`, `update()`, `deleteRecord()`, or `deleteMultiple()` method on bad query results can result in data loss.

You can set the `glide.invalid_query.returns_no_rows` system property to true to have queries with invalid encoded queries return no records.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>glideRecord</td>
<td>Object</td>
<td>GlideRecord from which to obtain the domain.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
//This example requires the Domain plugin be active, the Group table is the specified
//Domain table, and the ITIL user is in the Database Atlanta domain
//From any domain (using queryNoDomain()) look up the incidents that an ITIL user can only see
//who is in the Database Atlanta domain, should expect all incidents with the global or the
//Database Atlanta domain specified.
var domain = new GlideRecord('sys_user');
domain.addQuery('user_name', 'itil');
domain.queryNoDomain();
if (domain.next()) {
    var domainQuery = new GlideRecord('incident');
domainQuery.addDomainQuery(domain);
domainQuery.query();
gs.print('Number of Incidents for ITIL user: ' +
         domainQuery.getRowCount());
    while (domainQuery.next())
        gs.print(domainQuery.number);
}
```

Scoped equivalent

This method is not available in scoped applications.

GlideRecord - addEncodedQuery(String query)

Adds an encoded query to other queries that may have been set.

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an insert(), update(), deleteRecord(), or deleteMultiple() method on bad query results can result in data loss.

You can set the glide.invalid_query.returns_no_rows system property to true to have queries with invalid encoded queries return no records.

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>String</td>
<td>An encoded query string.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var queryString = "priority=1^ORpriority=2";
var gr = new GlideRecord('incident');
gr.addEncodedQuery(queryString);
gr.query();
while (gr.next()) {
    gs.addInfoMessage(gr.number);
}
```

Scoped equivalent

To use the `addEncodedQuery()` method in a scoped application, use the corresponding scoped method: `addEncodedQuery()`.  

**GlideRecord - addFunction(Object function)**

Applies a pre-defined GlideDBFunctionBuilder object to a record.

Use the GlideDBFunctionBuilder scoped class to define a function. After the function is defined, use the `addFunction(Object function)` method to apply the function to a record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>Object</td>
<td>A GlideDBFunctionBuilder object that defines a SQL operation.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var functionBuilder = new GlideDBFunctionBuilder();
var myAddingFunction = functionBuilder.add();
myAddingFunction = functionBuilder.field('order');
myAddingFunction = functionBuilder.field('priority');
myAddingFunction = functionBuilder.build();
```
var gr = new GlideRecord('incident');
gr.addFunction(myAddingFunction);
gr.addQuery(myAddingFunction, '<', 5);
gr.query();
while(gr.next())
gs.log(gr.getValue(myAddingFunction));

Output:

*** Script: 1
*** Script: 4
*** Script: 3
*** Script: 1
*** Script: 1
*** Script: 2
*** Script: 1
*** Script: 1

GlideRecord - addInactiveQuery()

Adds a filter to return inactive records. Inactive records have the active flag set to false.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryCondition</td>
<td>Records where the active flag is false.</td>
</tr>
</tbody>
</table>

var inc = new GlideRecord('incident');
inc.addInactiveQuery();
inc.query();

Scoped equivalent

In scoped applications use the scoped method addQuery("active","false").

GlideRecord - addJoinQuery(String table)

Adds a filter to return records based on a relationship in a related table.

For example, find all the users that are in the database group (users via sys_user_grmember table). Another example would be find all problems that have an assigned incident (problems via the incident.problem_id relationship).

This is not a true database join; rather, addJoinQuery() adds a subquery. So, while the result set is limited based on the join, the only fields that you have access to are those on the base table (those which are in the table with which the GlideRecord was initialized).
Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an `insert()`, `update()`, `deleteRecord()`, or `deleteMultiple()` method on bad query results can result in data loss.

You can set the `glide.invalid_query.returns_no_rows` system property to true to have queries with invalid encoded queries return no records.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>Table name</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryCondition</td>
<td>Records where the relationships match.</td>
</tr>
</tbody>
</table>

Find problems that have an incident attached. This example returns problems that have associated incidents. However, it won’t pull values from the incidents that are returned as a part of the query.

```javascript
var prob = new GlideRecord('problem');
prob.addJoinQuery('incident');
prob.query();
```

Find `active=false` problems with associated incidents.

```javascript
// Look for Problem records
var gr = new GlideRecord('problem');

// That have associated Incident records
var grSQ = gr.addJoinQuery('incident');

// Where the Problem records are "active=false"
gr.addQuery('active', 'false');

// And the Incident records are "active=true"
grSQ.addCondition('active', 'true');

// Query
gr.query();

// Iterate and print results
while (gr.next()) {
    gs.print(gr.getValue('number'));
}
```
**Scoped equivalent**

To use the `addJoinQuery()` method in a scoped application, use the corresponding scoped method: `addJoinQuery()`.

**GlideRecord - addJoinQuery(String table, String primaryField)**

Adds a filter to return records based on a relationship in a related table.

For example, find all the users that are in the database group (users via `sys_user_grmember` table). Another example would be find all problems that have an assigned incident (problems via the `incident.problem_id` relationship).

This is not a true database join; rather, `addJoinQuery()` adds a subquery. So, while the result set is limited based on the join, the only fields that you have access to are those on the base table (those which are in the table with which the GlideRecord was initialized).

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an `insert()`, `update()`, `deleteRecord()`, or `deleteMultiple()` method on bad query results can result in data loss.

You can set the `glide.invalid_query.returns_no_rows` system property to true to have queries with invalid encoded queries return no records.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>Table name</td>
</tr>
<tr>
<td>primaryField</td>
<td>String</td>
<td>If other than <code>sys_id</code>, the primary field.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryCondition</td>
<td>Records where the relationships match.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.

ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an `insert()`, `update()`, `deleteRecord()`, or `deleteMultiple()` method on bad query results can result in data loss.

You can set the `glide.invalid_query.returns_no_rows` system property to true to have queries with invalid encoded queries return no records.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>Table name</td>
</tr>
<tr>
<td>primaryField</td>
<td>String</td>
<td>If other than <code>sys_id</code>, the primary field.</td>
</tr>
<tr>
<td>joinTableField</td>
<td>String</td>
<td>If other than <code>sys_id</code>, the field that joins the tables</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryCondition</td>
<td>Records where the relationships match.</td>
</tr>
</tbody>
</table>

Find problems that have incidents associated where the incident `caller_id` field value matches that of the problem `opened_by` field.

```javascript
var gr = new GlideRecord('problem');
gr.addJoinQuery('incident', 'opened_by', 'caller_id');
gr.query();
```

### Scoped equivalent

To use the `addJoinQuery()` method in a scoped application, use the corresponding scoped method: `addJoinQuery0`.

### GlideRecord - `addNotNullQuery(String fieldName)`

Adds a filter to return records where the specified field is not null.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryCondition</td>
<td>QueryCondition of records where the parameter field is not null.</td>
</tr>
</tbody>
</table>

```javascript
var target = new GlideRecord('incident');
target.addNotNullQuery('short_description');
target.query();  // Issue the query to the database to get all records
while (target.next()) {
  // add code here to process the incident record
}
```

Scoped equivalent

To use the `addNotNullQuery()` method in a scoped application, use the corresponding scoped method: `addNotNullQuery()`.

GlideRecord - `addNullQuery(String fieldName)`

Adds a filter to return records where the specified field is null.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryCondition</td>
<td>QueryCondition of records where the specified field is null.</td>
</tr>
</tbody>
</table>

```javascript
var target = new GlideRecord('incident');
target.addNullQuery('short_description');
target.query();  // Issue the query to the database to get all records
while (target.next()) {
  // add code here to process the incident record
}
```
Scoped equivalent

To use the `addNullQuery()` method in a scoped application, use the corresponding scoped method: `addNullQuery()`.

GlideRecord - `addQuery(String name, Object operator, Object value)`

Provides the ability to build a request, which when executed, returns the rows from the specified table that match the request.

If you are familiar with SQL, this method is similar to the "where" clause. One or more `addQuery()` calls can be made in a single query; in this case the queries are AND'ed. If any of the query statements need to be OR'ed, use the class `GlideQueryCondition`.

`addQuery()` is typically called with three parameters; table field, operator, and comparison value. It can be called with only two parameters, table field and comparison value, such as `myObj.addQuery('category','Hardware');`. The operator in this case is assumed to be 'equal to'.

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an `insert()`, `update()`, `deleteRecord()`, or `deleteMultiple()` method on bad query results can result in data loss.

You can set the `glide.invalid_query.returns_no_rows` system property to true to have queries with invalid encoded queries return no records.

<p>| Parameters |
|-----------------|-----------------|-----------------|
| Name | Type | Description |
| name | String | Table field name |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| operator| Object    | Query operator. The available values are dependent on the data type of the value parameter. Numbers:  
  - =  
  - !=  
  - >  
  - >=  
  - <  
  - <=  
  Strings (must be in upper case):  
  - =  
  - !=  
  - IN  
  - NOT IN  
  - STARTSWITH  
  - ENDSWITH  
  - CONTAINS  
  - DOES NOT CONTAIN  
  - INSTANCEOF  

| value   | Object    | Value on which to query (not case-sensitive).                               |

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryCondition</td>
<td>A reference to the QueryCondition that was added to the GlideRecord.</td>
</tr>
</tbody>
</table>

```javascript
var rec = new GlideRecord('incident');
rec.addQuery('active',true);
rec.addQuery('sys_created_on', '>', "2010-01-19 04:05:00");
rec.query();
while (rec.next()) {
  rec.active = false;
  gs.print('Active incident ' + rec.number + ' closed');
  rec.update();
}
```

Using the IN operator.

```javascript
var que = new GlideRecord('incident');
que.addQuery('number','IN','INC00001,INC00002');
```
Scoped equivalent

To use the `addQuery()` method in a scoped application, use the corresponding scoped method: `addQuery()`.

GlideRecord - `applyTemplate(String template)`

Apply a template record from the Template table (sys_template) to the current record. If the specified template is not found, no action is taken.

**Note:** This method automatically instantiates a `gr.insert()` method if a template has the Next Related Child Template field filled. For information, see Create templates for related task records.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>template</td>
<td>String</td>
<td>Name of a template from the Templates (sys_template) table.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var rec1 = new GlideRecord("incident");
rec1.initialize();
rec1.applyTemplate("my_incident_template");
```

GlideRecord - `autoSysFields(Boolean e)`

Enables or disables the update to the fields sys_updated_by, sys_updated_on, sys_mod_count, sys_created_by, and sys_created_on. This is often used for manually updating field values on a record while leaving historical information unchanged.

**Note:** This is not available for scoped apps, starting with the Fuji release. See the Scoped GlideRecord API Reference for a list of what APIs are available for scoped apps.

**Caution:** Use caution if you use this method. When you use this method the sys_mod_count field will not be incremented, and other sys_fields will not be updated. This can break functionality including, but not limited to, the Activity Formatter, History Sets, Notifications, and Metrics.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>Boolean</td>
<td>If false disables updates to sys_updated_by, sys_updated_on, sys_mod_count, sys_created_by, and sys_created_on.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var inc = new GlideRecord('incident');
// Change all Open(1) incidents to Active(2)
inc.addQuery('state', 1);
inc.query();

while (inc.next()) {
    inc.autoSysFields(false); // Do not update sys_updated_by, sys_updated_on, sys_mod_count, sys_created_by, and sys_created_on
    inc.setWorkflow(false);    // Do not run any other business rules
    inc.setValue('state', 2);
    inc.update();
}
```

### GlideRecord - canCreate()

Determines if the access control rules (which includes the user's role) permit inserting new records in this table.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user's role permits creation of new records in the table.</td>
</tr>
</tbody>
</table>
Scoped equivalent

To use the `canCreate()` method in a scoped application, use the corresponding scoped method: `canCreate()`.

GlideRecord - canDelete()

Determines if the access control rules (which includes the user’s role) permit deletion of records in this table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user can delete records from this table, false otherwise.</td>
</tr>
</tbody>
</table>

```javascript
var att = new GlideRecord('sys_attachment');
att.get('[$sys_attachment.sys_id]');
var sm = GlideSecurityManager.get();
var checkMe = 'record/sys_attachment/delete';
var canDelete = sm.hasRightsTo(checkMe, att);
gs.log('canDelete: ' + canDelete);
```

Scoped equivalent

To use the `canDelete()` method in a scoped application, use the corresponding scoped method: `canDelete()`.

GlideRecord - canRead()

Determines if the access control rules (which includes the user’s role) permit reading this table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user can read from this table, false otherwise.</td>
</tr>
</tbody>
</table>
Scoped equivalent

To use the `canRead()` method in a scoped application, use the corresponding scoped method: `canRead()`.

**GlideRecord - `canWrite()`**

Determines if the access control rules (which includes the user’s role) permit updates to records in this table.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Boolean</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `canWrite()` method in a scoped application, use the corresponding scoped method: `canWrite()`.

**GlideRecord - `changes()`**

Determines whether any of the fields in the record have changed.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Boolean</td>
</tr>
</tbody>
</table>

Scoped equivalent

To implement this functionality in a scoped application, add code similar to the following:

```javascript
var gr = new GlideRecord("incident");
gr.get("965c9e5347c12200e0ef563d9b9a7156");
gr.short_description = "test";
var elements = gr.getElements();
```
var hasChanged = false;
for(var i=0; i < elements.length;i++){
    var element = elements[i];
    hasChanged = hasChanged || element.changes();
    gs.info(element.getName() + "":"" + element.changes());
}
gs.info(hasChanged);

**GlideRecord - deleteMultiple()**

Deletes multiple records according to the current "where" clause.

This method does not delete attachments.

Dot-walking is not supported for this method. When using the `deleteMultiple()` function on referenced tables, all the records in the table are deleted. Also, when using `deleteRecord()` to cascade delete, prior calls to `setWorkflow()` on the same GlideRecord object are ignored.

Do not use `deleteMultiple()` on tables with currency fields. Always delete each record individually. Also, do not use this method with the `chooseWindow()` or `setLimit()` methods when working with large tables.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```javascript
function nukeCart() {
    var cart = getCart();
    var id = cart.sys_id;
    var kids = new GlideRecord('sc_cart_item');
    kids.addQuery('cart', cart.sys_id);
    kids.deleteMultiple();
}
```

**Scoped equivalent**

To use the `deleteMultiple()` method in a scoped application, use the corresponding scoped method: `deleteMultiple()`.

**GlideRecord - deleteRecord()**

Deletes a single record.

ℹ️ **Note:** When using `deleteMultiple()` to cascade delete, prior calls to `setWorkflow()` on the same GlideRecord object are ignored.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Flag that indicates whether the record was successfully deleted.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>· true: Record was deleted.</td>
</tr>
<tr>
<td></td>
<td>· false: No record was found to delete.</td>
</tr>
</tbody>
</table>

Example

```javascript
var rec = new GlideRecord('incident');
rec.addQuery('active',false);
rec.query();
while (rec.next()) {
    gs.print('Inactive incident ' + rec.number + ' deleted');
    rec.deleteRecord();
}
```

Scoped equivalent

To use the `deleteRecord()` method in a scoped application, use the corresponding scoped method: `deleteRecord()`.

**GlideRecord - find(String columnName, String value)**

Returns true if any record has a matching value in the specified column. If found, it also moves to the first record that matches, essentially executing `next()` until the record is returned.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnName</td>
<td>String</td>
<td>Specifies the field name.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>Specifies the value to check for in the specified field.</td>
</tr>
</tbody>
</table>
## GlideRecord - get(Object name, Object value)

Returns the specified record in an instantiated GlideRecord object.

This method accepts either one or two parameters. If only a single parameter is passed in, the method assumes it is the sys_id of the desired record. If two parameters are passed in, the first is the name of the column within the instantiated GlideRecord to search. The second is the value for which to search.

If multiple records are found, use `next()` to access the additional records.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Object</td>
<td>Optional. Name of the instantiated GlideRecord column to search for the specified value parameter. If only a single parameter is passed in, the method assumes that this parameter is sys_id.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>Value to match.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Indicates whether the requested record was located:</td>
</tr>
<tr>
<td></td>
<td>• true: record was found</td>
</tr>
<tr>
<td></td>
<td>• false: record was not found</td>
</tr>
</tbody>
</table>

```javascript
var grIncident = new GlideRecord('incident');
var returnValue = 
grIncident.get('99ebb4156fa831005be8883e6b3ee4b9');
gs.info(returnValue); // logs true or false
 gs.info(grIncident.number); // logs Incident Number
```

```javascript
var grIncident = new GlideRecord('incident');
var returnValue = grIncident.get('caller_id.name','Sylivia Wayland');
gs.info(returnValue); // logs true or false
```
Scoped equivalent

To use the `get()` method in a scoped application, use the corresponding scoped method: `get()`.

GlideRecord - `getAttribute(String fieldName)`

Returns the dictionary attributes on the specified field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Field name for which to return the dictionary attributes</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Dictionary attributes</td>
</tr>
</tbody>
</table>

doit();

```javascript
function doit() {
  var gr = new GlideRecord('sys_user');
  gr.query("user_name","admin");
  if (gr.next()) {
    gs.print("we got one");
    gs.print(gr.location.getAttribute("tree_picker"));
  }
}
```

Scoped equivalent

To use the `getAttribute()` method in a scoped application, use the corresponding scoped method: `getAttribute()`.

GlideRecord - `getClassDisplayValue()`

Returns the table's label.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The table's label</td>
</tr>
</tbody>
</table>

### Scoped equivalent

To use the `getClassName()` method in a scoped application, use the corresponding scoped method: `getClassName()`.

### GlideRecord - `getDisplayValue()`

Retrieves the display value for the current record.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Display value for the current record</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gr.get('sys_id','ef43c6d40a0a0b5700c77f9bf387afe3');
gs.info(gr.getDisplayValue());
```

### GlideRecord - `getED()`

Returns the element's descriptor.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideElementDescriptor</td>
<td>The element's descriptor</td>
</tr>
</tbody>
</table>

```javascript
var totalCritical  = 0;
var filledCritical  = 0;
```
var fields = current.getFields();
gs.print(fields);
for (var num = 0; num < fields.size(); num++) {
gs.print("RUNNING ARRAY VALUE " + num);
var ed = fields.get(num).getED();
if(ed.hasAttribute("tiaa_critical")) {
gs.print("CRITICAL FIELD FOUND");
totalCritical ++;
if (!fields.get(num).isNil()) {
    filledCritical ++;
}
}
var answer = 0;
gs.print("TOTAL - " + totalCritical);
gs.print("FILLED - " + filledCritical);
if (filledCritical > 0 && totalCritical > 0) {
    var pcnt = (filledCritical/totalCritical)*100;
    answer = pcnt.toFixed(2);
}
answer;

Scoped equivalent

To use the getED() method in a scoped application, use the corresponding scoped method: getED().

GlideRecord - getElement(String fieldName)

Retrieves the GlideElement for a specified field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>A field name</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideElement</td>
<td>A GlideElement object</td>
</tr>
</tbody>
</table>
Scoped equivalent

To use the `getElement()` method in a scoped application, use the corresponding scoped method: `getElement()`.

**GlideRecord - getEncodedQuery()**

Retrieves the encoded query as a string.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `getEncodedQuery()` method in a scoped application, use the corresponding scoped method: `getEncodedQuery()`.

**GlideRecord - getEscapedDisplayValue()**

Retrieves the field value for the display field of the current record and adds escape characters for use in Jelly scripts.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

**GlideRecord - getFields()**

Retrieves a Java ArrayList of fields in the current record.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>
// This can be run in "Scripts - Background" for demonstration purposes

// Get a single incident record
var grINC = new GlideRecord('incident');
grINC.query();
grINC.next();
gs.print('Using ' + grINC.getValue('number'));
gs.print('');

// getFields() returns a Java ArrayList
var fields = grINC.getFields();

// Enumerate GlideElements in the GlideRecord object that have values
gs.print('Enumerating over all fields with values:');
for (var i = 0; i < fields.size(); i++) {
  var glideElement = fields.get(i);
  if (glideElement.hasValue()) {
    gs.print(' ' + glideElement.getName() + '	' +
      glideElement);
  }
}
gs.print('');

// Get a specific GlideElement: number
gs.print('Getting the number field:');
for (var i = 0; i < fields.size(); i++) {
  var glideElement = fields.get(i);
  if (glideElement.hasValue() && glideElement.getName() ==
    'number') {
    gs.print(' ' + glideElement.getName() + '	' +
      glideElement);
  }
}

GlideRecord - getLabel()
Retrieves the field’s label.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The field’s label</td>
</tr>
</tbody>
</table>

```
template.print("Summary of Requested items:\n");
var gr = new GlideRecord("sc_req_item");
gr.addQuery("request", current.sysapproval);
gr.query();
while(gr.next()) {
    var nicePrice = gr.price.toString();
    if (nicePrice !== '') {
        nicePrice = parseFloat(nicePrice);
        nicePrice = nicePrice.toFixed(2);
    }
    template.print(gr.number + ":  " + gr.quantity + " X " +
gr.cat_item.getDisplayValue()
        + " at $" + nicePrice + " each \n");
    template.print("    Options:\n");
    for (key in gr.variables) {
        var v = gr.variables[key];
        if(v.getGlideObject().getQuestion().getLabel() !== '') {
            template.space(4);
            template.print('     ' +
v.getGlideObject().getQuestion().getLabel() + " = "
                + v.getDisplayValue() + "\n");
        }
    }
}
```

Scoped equivalent

To use the `getLabel()` method in a scoped application, use the corresponding scoped method: `getLabel()`.

**GlideRecord - getLink(Boolean noStack)**

Retrieves the link for the current record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>noStack</td>
<td>Boolean</td>
<td>If true, the link generated will not append &amp;sysparm_stack=[tablename]_list.do?sysparm_query=active=true to the end of the URL; if false, the link will. Leaving the parameter empty defaults to false.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A URL</td>
</tr>
</tbody>
</table>

```javascript
//Check for attachments and add link if there are any
var attachment_link = '';  
var rec = new GlideRecord('sc_req_item');
rec.addQuery('sys_id', current.request_item);
rec.query();
if(rec.next()){
    if(rec.hasAttachments()){
        attachment_link = gs.getProperty('glide.servlet.uri') + rec.getLink();
    }
}
```

Scoped equivalent

To use the `getLink()` method in a scoped application, use the corresponding scoped method: `getLink0`.

**GlideRecord - getLocation(Boolean b)**

Retrieves the current row number.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The row number of the current record</td>
</tr>
</tbody>
</table>

**GlideRecord - getPlural()**

Retrieves the plural label of the GlideRecord table.

For example, if the table name is "Change Request," this method returns "Change Requests."

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The plural label of the GlideRecord’s table.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `getPlural()` method in a scoped application, use the scoped `GlideRecord.getED()` method to obtain the field’s descriptor, and then use the scoped `GlideElementDescriptor.getPlural()` method.

```javascript
lnr.query();
lnr.next();
var ed = lnr.getED();
gs.info(ed.getPlural());
```

GlideRecord - `getRecordClassName()`

Retrieves the class (table) name for the current record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Class or table name</td>
</tr>
</tbody>
</table>

```javascript
function TaskAssignmentFilter() {
    var classname = current.getRecordClassName();
    var filter = "type=null";
    if (classname == "incident" && current.category == "database") {
        filter = GetGroupFilter("database");
    }
    else {
        // append exclusion for 'catalog' to the filter
        var cat = new GlideRecord("sys_user_group_type");
        cat.addQuery("name", "catalog");
        cat.query();
        if (cat.next()) {
            filter += "^ORtype!=" + cat.sys_id;
        }
    }
    gs.log("TaskAssignmentFilter: " + filter);
    return filter;
}
```
Scoped equivalent

To use the `getRecordClassName()` method in a scoped application, use the corresponding scoped method: `getRecordClassName()`.

GlideRecord - `getRelatedLists(Boolean b)`
Retrieves a list of names and display values of tables that refer to the current record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HashMap</td>
<td>Hash map with names and display values of related tables.</td>
</tr>
</tbody>
</table>

GlideRecord - `getRelatedTables(Boolean b)`
Retrieves a list of names and display values of tables that are referred to by the current record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HashMap</td>
<td>Hash map with names and display values of related tables.</td>
</tr>
</tbody>
</table>

GlideRecord - `getRowCount()`
Retrieves the number of rows in the GlideRecord object.

**Note:** This method should not be used in a production environment as it creates a heavy load on the system. Instead, use `GlideAggregate()`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Number of rows</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `getRowCount()` method in a scoped application, use the corresponding scoped method: `getRowCount()`.

GlideRecord - `getRowNumber()`

Retrieves the row number set by `saveLocation()` or `setLocation()`.

To get the current row number, use `getLocation()`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The saved row number</td>
</tr>
</tbody>
</table>

GlideRecord - `getTableName()`

Retrieves the table name associated with this GlideRecord.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A table name</td>
</tr>
</tbody>
</table>

```javascript
gs.log('Table: ' + current.getTableName());
gs.log('Parent: ' + current.parent.sys_id);
var item = new GlideRecord('sc_req_item');
item.addQuery('sys_id', current.parent.sys_id);
item.query();
if(item.next()){
  for(var variable in item.variable_pool) { gs.log(variable);
```
```javascript
var answer = eval("item.variable_pool." + variable + ".getDisplayValue()";
gs.log(answer);
}
```

**Scoped equivalent**

To use the `getTableName()` method in a scoped application, use the corresponding scoped method: `getTableName()`.

**GlideRecord - getValue(String fieldName)**

Retrieves the string value of an underlying element in a field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of a field</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The string value of the underlying element. Returns null if the field is empty or the field does not exist. Boolean values return as &quot;0&quot; and &quot;1&quot; string values instead of false and true.</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `getValue()` method in a scoped application, use the corresponding scoped method: `getValue()`.

**GlideRecord - GlideRecord(String tableName)**

Creates an instance of the GlideRecord class for the specified table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table to be used.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
```

**GlideRecord - hasAttachments()**

Determines if the current record has any attachments.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the current record has attachments, false otherwise.</td>
</tr>
</tbody>
</table>

```java
//Check for attachments and add link if there are any
var attachment_link = '';
var rec = new GlideRecord('sc_req_item');
rec.addQuery('sys_id', current.request_item);
rec.query();
if(rec.next()){
   if(rec.hasAttachments()){
      attachment_link = gs.getProperty('glide.servlet.uri') + rec.getLink();
   }
}
```

**GlideRecord - hasNext()**

Determines if there are any more records in the GlideRecord.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if there are more records in the query set.</td>
</tr>
</tbody>
</table>

```java
if (gr.hasNext()) {
   dothis(); // found it, do it
} else {
   dothat(); // didn't find it
}
```
Scoped equivalent

To use the `hasNext()` method in a scoped application, use the corresponding scoped method: `hasNext()`.

GlideRecord - `initialize()`

Creates an empty record suitable for population before an insert.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('to_do');
gr.initialize();
gr.name = 'first to do item';
gr.description = 'learn about GlideRecord';
gr.insert();
```

Scoped equivalent

To use the `initialize()` method in a scoped application, use the corresponding scoped method: `initialize()`.

GlideRecord - `insert()`

Inserts a new record using the field values that have been set for the current record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the inserted record, or null if the record is not inserted.</td>
</tr>
</tbody>
</table>
var gr = new GlideRecord('to_do');
gr.initialize();
gr.name = 'first to do item';
gr.description = 'learn about GlideRecord';
gr.insert();

Scoped equivalent

To use the `insert()` method in a scoped application, use the corresponding scoped method: `insert()`.

**GlideRecord - insertWithReferences()**

Inserts a new record and also inserts or updates any related records with the provided information.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>sys_id of the inserted record or null if the record was not inserted.</td>
</tr>
</tbody>
</table>

If a reference value is not specified (as below), then a new user record is created with the provided first_name and last_name, and the caller_id value is set to this newly created sys_user record. The result is a new sys_user record with the provided first_name and last_name and a new incident record with the provided short_description and caller_id.

```
var inc = new GlideRecord('incident');
inc.initialize();
inc.short_description = 'New incident 1';
inc.caller_id.first_name = 'John';
inc.caller_id.last_name = 'Doe';
inc.insertWithReferences();
```

If a caller_id value is specified, then that caller_id is updated with the provided first_name and last_name. The result is a newly created incident record with values set for short_description and caller_id.

```
var inc = new GlideRecord('incident');
inc.initialize();
inc.short_description = 'New incident 1';
inc.caller_id.setDisplayValue('David Loo');
inc.caller_id.first_name = 'John';
```
GlideRecord - instanceof(String className)

Checks a table for the type/class of table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>className</td>
<td>String</td>
<td>Name of a type or class of record.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if table is an instance of the specified class.</td>
</tr>
</tbody>
</table>

GlideRecord - isNewRecord()

Determines whether the current record has been inserted into the database. This method returns true only if the newRecord() method has been called. This method is useful for scripted ACL, and in the condition of UI actions, but should not be used in background scripts.

**Note:** This method returns true for any new record during a business rule, or if the newRecord() method is used to initialize a record with default values and a unique ID (sys_id). In all other cases, it returns false.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the current record is new (has not been inserted into the database.)</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the isNewRecord() method in a scoped application, use the corresponding scoped method: isNewRecord().

GlideRecord - isValid()

Determines if the table exists.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

`gs.print(testTable.isValid());`

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the table is valid or if the record was successfully fetched, false otherwise.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `isValid()` method in a scoped application, use the corresponding scoped method: `isValid()`.

**GlideRecord - `isValidField(String fieldName)`**

Determines if the specified field is defined in the current table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of a field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field is valid, false otherwise.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `isValidField()` method in a scoped application, use the corresponding scoped method: `isValidField()`.

**GlideRecord - `isValidRecord()`**

Determines if the current record is valid.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the current record is valid or false if past the end of the record set.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `isValidRecord()` method in a scoped application, use the corresponding scoped method: `isValidRecord()`.

GlideRecord - `newRecord()`

Creates a GlideRecord, set the default values for the fields and assign a unique id to the record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `newRecord()` method in a scoped application, use the corresponding scoped method: `newRecord()`.

GlideRecord - `next()`

Moves to the next record in the GlideRecord.

Use this method to iterate through the records returned by a GlideRecord query.

**Note:** This method fails if there is a field in the table called "next". If that is the case, use the method `_next()`.

**Note:** The `if(myObj.next())` construct only processes the first record returned.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Boolean| Flag that indicates if there is a "next" record in the GlideRecord. Valid values:  
· true: Move to the next record was successful.  
· false: No more records in the result set. |

Example

```javascript
var rec = new GlideRecord('incident');
rec.query();
while (rec.next()) {
    gs.print(rec.number + ' exists');
}
```

Scoped equivalent

To use the `next()` method in a scoped application, use the corresponding scoped method: `next()`.

GlideRecord - `_next()`

Moves to the next record in the GlideRecord. Provides the same functionality as `next()`, intended to be used in cases where the GlideRecord has a column named `next`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if there are more records in the query set.</td>
</tr>
</tbody>
</table>
Scoped equivalent

To use the `_next()` method in a scoped application, use the corresponding scoped method: `_next()`.

```javascript
var rec = new GlideRecord('sys_template');
rec.query();
while (rec._next()) {
    gs.print(rec.number + ' exists');
}
```

GlideRecord - operation()

Retrieves the current operation being performed, such as insert, update, delete, etc.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current operation</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `operation()` method in a scoped application, use the corresponding scoped method: `operation()`.

GlideRecord - orderBy(String fieldName)

Specifies a field name to be used to order the query set. This may be called more than once to order by multiple fields.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>A field name</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
function UpdateProjectWBS(project) {
    // Code goes here
}
```
var count = 0;
var child = new GlideRecord('pm_project_task');
child.addQuery('parent', project.sys_id);
child.orderBy('order');
child.orderBy('number');
child.query();
var len = child.getRowCount().toString().length;
var seq = 0;
while (child.next()) {
    count += UpdateProjectTaskWBS(child, 1, ++seq, len, '');
}
gs.addInfoMessage(count + ' Project Tasks updated');

Scoped equivalent

To use the `orderBy()` method in a scoped application, use the corresponding scoped method: `orderBy()`.  

GlideRecord - orderByDesc(String, fieldName)

Specifies a field used to order the query set in descending order.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>A field name.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `orderByDesc()` method in a scoped application, use the corresponding scoped method: `orderByDesc()`.  

GlideRecord - query(Object field, Object value)

Runs the query against the table based on the filters specified by `addQuery()` and `addEncodedQuery()`.

This method queries the GlideRecord table as well as any references of the table. One argument adds a query string. Usually this is performed without arguments, but you can specify a name/value pair.

Note: This method fails if there is a field in the table called "query". If that is the case, use the method `_query()`.  

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>Object</td>
<td>Field name</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>Field value</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var rec = new GlideRecord('incident');
rec.query();
while (rec.next()) {
    gs.print(rec.number + ' exists');
}
```

Scoped equivalent

To use the `query()` method in a scoped application, use the corresponding scoped method: `query()`.

**GlideRecord - _query(Object field, Object value)**

Identical to `query()`. This method is intended to be used on tables where there is a column named query, which would interfere with using the `query()` method.

Runs the query against the table based on the filters specified by the `addQuery()` and `addEncodedQuery()` methods. This will query the GlideRecord table as well as any references of the table. One argument adds a query string. Usually this is performed without arguments, but a name/value pair can be specified.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Object</td>
<td>A field name</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>A value</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
var rec = new GlideRecord('sys_app_module');
rec._query();
while (rec.next()) {
  gs.print(rec.number + ' exists');
}

Scoped equivalent

To use the _query() method in a scoped application, use the corresponding scoped method: _query().

GlideRecord - queryNoDomain(Object field, Object value)

Used in domain separated instances. Similar to query(), runs the query against the table based on the filters specified by addQuery() and addEncodedQuery(), but ignores domains.

This will query the GlideRecord table as well as any references of the table. One argument adds a query string. Usually this is performed without arguments, but a name/value pair can be specified.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>Object</td>
<td>A field name</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>A value</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideRecord - restoreLocation()

Sets the current record to be the record that was saved with saveLocation(). If saveLocation() has not been called, the current record is set to be the first record of the GlideRecord.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GlideRecord - saveLocation()
Saves the current row number so that we can get back to this location using the
restoreLocation() method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GlideRecord - setAbortAction(Boolean b)
Sets a flag to indicate if the next database action (insert, update, delete) is to be aborted.
Use in an onBefore business rule to prevent the database action from being done. The business
rule continues to run after setAbortAction() is called. Calling setAbortAction() does not stop
subsequent business rules from executing. Calling this method only prevents the database action
from occurring.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>True to abort next action, or false to allow the next action.</td>
</tr>
</tbody>
</table>

if ( (!current.u_date1.nil()) && (!current.u_date2.nil()) ) {
    var start =
    current.u_date1.getGlideObject().getNumericValue();
    var end = current.u_date2.getGlideObject().getNumericValue();
    if (start > end) {
        gs.addInfoMessage('start must be before end');
        current.u_date1.setError('start must be before end');
        current.setAbortAction(true);
GlideRecord - setDisplayValue(String name, Object value)

Sets the specified field to the specified display value.
For a reference field this is the display value for the table. For a date/time this is the time in the
caller's current timezone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Field name</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>Display value for the specified field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gr.get('46f09e75a9fe198100f4ffd8d366d17b');
gr.setDisplayValue('opened_at','2011-02-13 4:30:00');
gr.update();
```

GlideRecord - setForceUpdate(Boolean force)

Updates the record even if fields have not changed.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>force</td>
<td>Boolean</td>
<td>True to update even if fields have not changed, otherwise false.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideRecord - setLimit(Number limit)

Sets the limit for how many records are in the GlideRecord.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>Number</td>
<td>Limit for records to fetch.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gr.orderByDesc('sys_created_on');
gr.setLimit(10);
gr.query();
```

Scoped equivalent

To use the `setLimit()` method in a scoped application, use the corresponding scoped method: `setLimit()`.

GlideRecord - setLocation(Number rowNumber)

Sets the current row location.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rowNumber</td>
<td>Number</td>
<td>The row number to set as the current row.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideRecord - setNewGuid()

Generates a new GUID and sets it as the unique id for the current record. This function applies only to new records. The GUID for an existing record cannot be changed.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ServiceNow Kingston Now Platform Custom Business Applications

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var tsk_id = task.setNewGuid();
task.description = "Request: " + current.request.number;
task.description = task.description + "\n" + "Requested by: " +
current.request.u_requested_by.name;
task.description = task.description + "\n" + "Requested for: " +
current.request.u_requested_for.name;
task.description = task.description + "\n" + "Item: " +
current.cat_item.name;

var gr = new GlideRecord ('task_rel_task');
//link the incident to the request (may need to review if it
needs to be the item)
gr.parent = current.request;
gr.child = tsk_id;
gr.insert();
```

**GlideRecord - setNewGuidValue (String guid)**

Generates a new GUID and sets it as the unique id for the current record, when inserting a new record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>guid</td>
<td>String</td>
<td>A string value for the new GUID</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `setNewGuidValue()` method in a scoped application, use the corresponding scoped method: `setNewGuidValue()`.

**GlideRecord - setQueryReferences(Boolean queryReferences)**

Enables or disables using the reference field's display name when querying a reference field.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryReferences</td>
<td>Boolean</td>
<td>If true, will generate a string of display names. If false, will generate a string of sys_ids.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideRecord - setUseEngines(Boolean e)**

Disables or enables the running of any engines (approval rules / assignment rules).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>Boolean</td>
<td>If true, enables engines. If false disables engines.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideRecord - setValue(String name, Object value)**

Sets the specified field to the specified value.

Normally a script would do a direct assignment, for example, `gr.category = value`. However, if in a script the element name is a variable, then `gr.setValue(elementName, value)` can be used. When setting a value, ensure the data type of the field matches the data type of the value you enter. This method cannot be used on journal fields.

If the value parameter is null, the record is not updated, and an error is not thrown.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Field name</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>A value to be assigned.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
Scoped equivalent

To use the `setValue()` method in a scoped application, use the corresponding scoped method: `setValue()`.

**GlideRecord - setWorkflow(Boolean e)**

Enables or disables the running of business rules that might normally be triggered by subsequent actions. If the `e` parameter is set to false, an insert/update will not be audited. Auditing only happens when the parameter is set to true for a GlideRecord operation.

**Note:** The `setWorkflow()` method is ignored when subsequently using either the `deleteProblem()` or `deleteMultiple()` methods to cascade delete.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>Boolean</td>
<td>If true (default), enables business rules. If false, disables business rules.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

doit('name1','name2');

```javascript
function doit(username1, username2) {
    var usr1 = new GlideRecord('sys_user');
    var usr2 = new GlideRecord('sys_user');
    var num = 0;

    if (usr1.get('user_name', username1) &&
        usr2.get('user_name', username2)) {
        var ref;
        var dict = new GlideRecord('sys_dictionary');
        dict.addQuery('reference', 'sys_user');
        dict.addQuery('internal_type', 'reference');
        dict.query();
        while (dict.next()) {
            num = 0;
            ref = new GlideRecord(dict.name.toString());
            ref.addQuery(dict.element, usr1.sys_id);
            ref.query();
            while (ref.next()) {
                ref.setValue(dict.element.toString(), usr2.sys_id);
                ref.setWorkflow(false);
                ref.update();
                num++;
            }
        }
    }
    if (num > 0) {
    }
```
Scoped equivalent

To use the `setWorkflow()` method in a scoped application, use the corresponding scoped method: `setWorkflow()`.

GlideRecord - `update(Object reason)`

Updates the GlideRecord with any changes that have been made. If the record does not exist, it is inserted.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reason</td>
<td>Object</td>
<td>Optional. Reason for the update. The reason appears in the audit record.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the new or update record. Returns null if the update fails.</td>
</tr>
</tbody>
</table>

**Example**

```javascript
var gr = new GlideRecord('task_ci');
gr.addQuery();
gr.query();
var count = gr.getRowCount();
if (count > 0) {
    var allocation = parseInt(10000 / count) / 100;
    while (gr.next()) {
        gr.u_allocation = allocation;
        gr.update();
    }
}
```

Scoped equivalent

To use the `update()` method in a scoped application, use the corresponding scoped method: `update()`.
**GlideRecord - updateMultiple()**

Updates each GlideRecord in a stated query with a specified set of changes.

When changing field values, use `setValue()` instead of directly setting the field (`field = something`). When using `updateMultiple()`, directly setting the field (`gr.state = 4`) results in all records in the table being updated instead of just the records returned by the query.

Do not use this method with the `chooseWindow()` or `setLimit()` methods when working with large tables.

This method sets new values and does not clear existing values.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
// update the state of all active incidents to 4 - "Awaiting User Info"
var gr = new GlideRecord('incident');
gr.addQuery('active', true);
gr.setValue('state', 4);
gr.updateMultiple();
```

**Scoped equivalent**

To use the `updateMultiple()` method in a scoped application, use the corresponding scoped method: `updateMultiple()`.

**GlideRecord - updateWithReferences(Object reason)**

Updates a record and also inserts or updates any related records with the information provided.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reason</td>
<td>Object</td>
<td>A string designating the reasons for the updates. The reason is displayed in the audit record.</td>
</tr>
</tbody>
</table>
### GlideRecord

A GlideRecord contains both records and fields. For information about GlideRecordSecure, which is a class inherited from GlideRecord that performs the same functions as GlideRecord, and also enforces ACLs, see the GlideServer APIs.

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an insert(), update(), deleteRecord(), or deleteMultiple() method on bad query results can result in data loss.

You can set the glide.invalid_query.returns_no_rows system property to true to have queries with invalid encoded queries return no records.

### Scoped GlideRecord - addActiveQuery()

Adds a filter to return active records.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryCondition</td>
<td>Filter to return active records.</td>
</tr>
</tbody>
</table>

```javascript
var inc = new GlideRecord('incident');
inc.addActiveQuery();
inc.query();
```

**Scoped GlideRecord - addEncodedQuery(String query)**

Adds an encoded query to other queries that may have been set.

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an insert(), update(), deleteRecord(), or deleteMultiple() method on bad query results can result in data loss.

You can set the glide.invalid_query.returns_no_rows system property to true to have queries with invalid encoded queries return no records.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>String</td>
<td>An encoded query string.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```javascript
var queryString = "priority=1^ORpriority=2";
var gr = new GlideRecord('incident');
gr.addEncodedQuery(queryString);
gr.query();
while (gr.next()) {
    gs.addInfoMessage(gr.number);
}
```

**Scoped GlideRecord - addFunction(Object function)**

Applies a pre-defined GlideDBFunctionBuilder object to a record.

Use the GlideDBFunctionBuilder scoped class to define a function. After the function is defined, use the addFunction(Object function) method to apply the function to a record.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>Object</td>
<td>A GlideDBFunctionBuilder object that defines a SQL operation.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var functionBuilder = new GlideDBFunctionBuilder();
var myAddingFunction = functionBuilder.add();
myAddingFunction = functionBuilder.field('order');
myAddingFunction = functionBuilder.field('priority');
myAddingFunction = functionBuilder.build();

var gr = new GlideRecord('incident');
gr.addFunction(myAddingFunction);
gr.addQuery(myAddingFunction, '<', 5);
gr.query();
while(gr.next())
    gs.log(gr.getValue(myAddingFunction));
```

Output:

```text
*** Script: 1
*** Script: 4
*** Script: 3
*** Script: 1
*** Script: 1
*** Script: 2
*** Script: 1
*** Script: 1
```

**Scoped GlideRecord - addJoinQuery(String joinTable, Object primaryField, Object joinTableField)**

Adds a filter to return records based on a relationship in a related table.

You can use this method to find all the users that are in the database group via the [sys_user_grmember] table, or to find all problems that have an assigned incident via the incident.problem_id relationship.

This is not a true database join; rather, addJoinQuery adds a subquery. So, while the result set is limited based on the join, the only fields that you have access to are those on the base table (those which are in the table with which the GlideRecord was initialized).

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an insert(), update(), deleteRecord(), or deleteMultiple() method on bad query results can result in data loss.

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
You can set the glide.invalid_query.returns_no_rows system property to true to have queries with invalid encoded queries return no records.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>joinTable</td>
<td>String</td>
<td>Table name</td>
</tr>
<tr>
<td>primaryField</td>
<td>Object</td>
<td>(Optional) If other than sys_id, the primary field</td>
</tr>
<tr>
<td>joinTableField</td>
<td>Object</td>
<td>(Optional) If other than sys_id, the field that joins the tables</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>A filter that lists records where the relationships match.</td>
</tr>
</tbody>
</table>

Find problems that have an incident attached. This example returns problems that have associated incidents. However, it won’t pull values from the incidents that are returned as a part of the query.

```javascript
var prob = new GlideRecord('problem');
prob.addJoinQuery('incident');
prob.query();
```

Find inactive problems with associated incidents

```javascript
// Look for Problem records that have associated Incident records
var gr = new GlideRecord('problem');
var grSQ = gr.addJoinQuery('incident');

// Where the Problem records are "active=false"
gr.addQuery('active', 'false');

// And the Incident records are "active=true"
grSQ.addCondition('active', 'true');

// Query
gr.query();

// Iterate and output results
while (gr.next()) {
    gs.info(gr.getValue('number'));
}
```
Find problems that have incidents associated where the incident caller_id field value matches that of the problem opened_by field.

```java
var gr = new GlideRecord('problem');
gr.addJoinQuery('incident', 'opened_by', 'caller_id');
gr.query();
```

**Scoped GlideRecord - addNotNullQuery(String fieldName)**

A filter that specifies records where the value of the field passed in the parameter is not null.

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an `insert()`, `update()`, `deleteRecord()`, or `deleteMultiple()` method on bad query results can result in data loss.

You can set the `glide.invalid_query.returns_no_rows` system property to true to have queries with invalid encoded queries return no records.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field to check.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>Filter that specifies records where the value of the field passed in the parameter is not null.</td>
</tr>
</tbody>
</table>

**Example**

```java
var target = new GlideRecord('incident');
target.addNotNullQuery('short_description');
target.query();  // Issue the query to the database to get all records where short_description is not null
while (target.next()) {
  // add code here to process the incident record
}
```

**Scoped GlideRecord - addNullQuery(String fieldName)**

Adds a filter to return records where the value of the specified field is null.

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an `insert()`, `update()`, `deleteRecord()`, or `deleteMultiple()` method on bad query results can result in data loss.
You can set the glide.invalid_query.returns_no_rows system property to true to have queries with invalid encoded queries return no records.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field to check.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>Query condition added to the GlideRecord.</td>
</tr>
</tbody>
</table>

### Example

```javascript
var target = new GlideRecord('incident');
target.addNullQuery('short_description');
target.query();  // Issue the query to the database to get all records where short_description is null
while (target.next()) {
  // add code here to process the incident record
}
```

### Scoped GlideRecord - addQuery(String query)

Adds a filter to return records using an encoded query string.

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an insert(), update(), deleteRecord(), or deleteMultiple() method on bad query results can result in data loss.

You can set the glide.invalid_query.returns_no_rows system property to true to have queries with invalid encoded queries return no records.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>String</td>
<td>An encoded query string.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>Query condition added to the GlideRecord.</td>
</tr>
</tbody>
</table>
Example

```javascript
var rec = new GlideRecord('incident');
rec.addQuery('active=true');
rec.query();
while (rec.next()) {
    rec.active = false;
    gs.info('Active incident ' + rec.number + ' closed');
    rec.update();
}
```

Scoped GlideRecord - `addQuery(String name, Object value)`

Provides the ability to build a request, which when executed, returns the rows from the specified table, that match the request.

If you are familiar with SQL, this method is similar to the "where" clause. One or more `addQuery()` calls can be made in a single query; in this case the queries are AND'ed. If any of the query statements need to be OR'ed, use the `GlideQueryCondition` method `addOrCondition()`.

When `addQuery()` is called with only two parameters, table field and comparison value, such as `myObj.addQuery('category','Hardware');`, the operator in this case is assumed to be "equal to".

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an `insert()`, `update()`, `deleteRecord()`, or `deleteMultiple()` method on bad query results can result in data loss.

You can set the glide.invalid_query.returns_no_rows system property to true to have queries with invalid encoded queries return no records.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Table field name.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>Value on which to query (not case-sensitive).</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>Query condition added to the GlideRecord.</td>
</tr>
</tbody>
</table>

Example

```javascript
var rec = new GlideRecord('incident');
rec.addQuery('active', true);
rec.query();
while (rec.next()) {
    rec.active = false;
    gs.info('Active incident ' + rec.number + ' closed');
    rec.update();
}
```
Scoped GlideRecord - addQuery(String name, String operator, Object value)

Provides the ability to build a request, which when executed, returns the rows from the specified table, that match the request.

If you are familiar with SQL, this method is similar to the "where" clause. One or more addQuery() calls can be made in a single query; in this case the queries are AND'ed. If any of the query statements need to be OR'ed, use the GlideQueryCondition method addOrCondition().

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an insert(), update(), deleteRecord(), or deleteMultiple() method on bad query results can result in data loss.

You can set the glide.invalid_query.returns_no_rows system property to true to have queries with invalid encoded queries return no records.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Table field name.</td>
</tr>
<tr>
<td>operator</td>
<td>String</td>
<td>Query operator. The available values are dependent on the data type of the value parameter. Numbers:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• =</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• !=</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &gt;=</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &lt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &lt;=</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>Value on which to query (not case-sensitive).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Strings (must be in upper case):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• =</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• !=</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• NOT IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STARTSWITH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ENDSWITH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CONTAINS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DOES NOT CONTAIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• INSTANCEOF</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideQueryCondition</td>
<td>The query condition that was added to the GlideRecord.</td>
</tr>
</tbody>
</table>

Example

```javascript
var rec = new GlideRecord('incident');
rec.addQuery('active',true);
rec.addQuery('sys_created_on', '>', "2010-01-19 04:05:00");
rec.query();
while (rec.next()) {
    rec.active = false;
    gs.info('Active incident ' + rec.number + ' closed');
    rec.update();
}
```

Using the IN operator.

```javascript
var gr = new GlideRecord('incident');
gr.addQuery('number','IN','INC00001,INC00002');
gr.query();
while(gr.next()) {
    //do something....
}
```

Scoped GlideRecord - canCreate()

Determines if the Access Control Rules, which include the user's roles, permit inserting new records in this table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Flag that indicates whether the user's roles permit creating of records in this table.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>• true: Create permitted</td>
</tr>
<tr>
<td></td>
<td>• false: Create is not permitted</td>
</tr>
</tbody>
</table>
Example

```javascript
var gr = new GlideRecord('incident');
gs.info(gr.canCreate());
```

**Scoped GlideRecord - canDelete()**

Determines if the Access Control Rules, which include the user's roles, permit deleting records in this table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Flag that indicates whether the user's roles permit deleting of records in this table.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>· true: Deleting permitted</td>
</tr>
<tr>
<td></td>
<td>· false: Deleting is not permitted</td>
</tr>
</tbody>
</table>

Example

```javascript
var att = new GlideRecord('sys_attachment');
gs.info(att.canDelete());
```

**Scoped GlideRecord - canRead()**

Determines if the Access Control Rules, which include the user's roles, permit reading records in this table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Flag that indicates whether the user’s roles permit reading of records in this table.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>· true: Reading permitted</td>
</tr>
<tr>
<td></td>
<td>· false: Reading is not permitted</td>
</tr>
</tbody>
</table>

Example

```javascript
var gr = new GlideRecord('incident');
gs.info(gr.canRead());
```

Scoped GlideRecord - canWrite()

Determines if the Access Control Rules, which include the user’s roles, permit editing records in this table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Flag that indicates whether the user’s roles permit writing of records in this table.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>· true: Writing permitted</td>
</tr>
<tr>
<td></td>
<td>· false: Writing is not permitted</td>
</tr>
</tbody>
</table>

Example

```javascript
var gr = new GlideRecord('incident');
gs.info(gr.canWrite());
```

Scoped GlideRecord - chooseWindow(Number firstRow, Number lastRow, Boolean forceCount)

Sets a range of rows to be returned by subsequent queries.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstRow</td>
<td>Number</td>
<td>The first row to include. Because the index starts at 0, a value of 0 returns the first row.</td>
</tr>
<tr>
<td>lastRow</td>
<td>Number</td>
<td>The last row to include.</td>
</tr>
<tr>
<td>forceCount</td>
<td>Boolean</td>
<td>If true, the getRowCount() method will return all possible records.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gr.orderBy('number');
gr.chooseWindow(2, 4);
gr.query();
if (gr.next()) {
    gs.info(gr.number + ' is within window');
}
```

Scoped GlideRecord - deleteMultiple()  

Deletes multiple records that satisfy the query condition. This method does not delete attachments.  

Do not use `deleteMultiple()` on tables with currency fields. Always delete each record individually. Also, do not use this method with the `chooseWindow()` or `setLimit()` methods when working with large tables.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var gr = new GlideRecord('incident');
```
gr.addQuery('active','false'); //to delete all inactive incidents
gr.query();
gr.deleteMultiple();

Scoped GlideRecord - deleteRecord()
Deletes the current record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Flag that indicates whether the record was successfully deleted.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>· true: Record was deleted.</td>
</tr>
<tr>
<td></td>
<td>· false: No record was found to delete.</td>
</tr>
</tbody>
</table>

Example

```javascript
var gr = new GlideRecord('incident');
//to delete one record
if (gr.get('99ebb4156fa831005be8883e6b3ee4b9'))
    gr.deleteRecord();
```

Scoped GlideRecord - get(Object name, Object value)
Returns the specified record in an instantiated GlideRecord object.

This method accepts either one or two parameters. If only a single parameter is passed in, the method assumes it is the sys_id of the desired record. If two parameters are passed in, the first is the name of the column within the instantiated GlideRecord to search. The second is the value for which to search.

If multiple records are found, use next() to access the additional records.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Object</td>
<td>Optional. Name of the instantiated GlideRecord column to search for the specified value parameter. If only a single parameter is passed in, the method assumes that this parameter is sys_id.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>Value to match.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Indicates whether the requested record was located:</td>
</tr>
<tr>
<td></td>
<td>· true: Record was found</td>
</tr>
<tr>
<td></td>
<td>· false: Record was not found</td>
</tr>
</tbody>
</table>

Example using sys_id.

```javascript
var grIncident = new GlideRecord('incident');
var returnValue = grIncident.get('99ebb4156fa831005be8883e6b3ee4b9');
gs.info(returnValue); // logs true or false
gs.info(grIncident.number); // logs Incident Number
```

Example using the name of the column within the instantiated GlideRecord to search and the value for which to search.

```javascript
var grIncident = new GlideRecord('incident');
var returnValue = grIncident.get('caller_id.name','Sylivia Wayland');
gs.info(returnValue); // logs true or false
gs.info(grIncident.number); // logs Incident Number
```

Scoped GlideRecord - getAttribute(String fieldName)

Returns the dictionary attributes for the specified field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Field name for which to return the dictionary attributes</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Dictionary attributes</td>
</tr>
</tbody>
</table>

```
doit();
function doit() {
  var gr = new GlideRecord('sys_user');
  gr.query("user_name","admin");
  if (gr.next()) {
    gs.print("we got one");
    gs.print(gr.location.getAttribute("tree_picker"));
  }
}
```

Scoped GlideRecord - getClassDisplayValue()

Returns the table’s label.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Table’s label</td>
</tr>
</tbody>
</table>

Scoped GlideRecord - getDisplayValue()

Retrieves the display value for the current record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The display value for the current record.</td>
</tr>
</tbody>
</table>

```
var gr = new GlideRecord('incident');
gr.get('sys_id','ef43c6d40a0a0b5700c77f9bf387afe3');
```
Scoped GlideRecord - getED()

Returns the element's descriptor.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideElementDescriptor</td>
<td>Element's descriptor</td>
</tr>
</tbody>
</table>

```javascript
var totalCritical = 0;
var filledCritical = 0;
var fields = current.getFields();
gs.print(fields);
for (var num = 0; num < fields.size(); num++) {
gs.print("RUNNING ARRAY VALUE " + num);
var ed = fields.get(num).getED();
if(ed.hasAttribute("tiaa_critical")) {
gs.print("CRITICAL FIELD FOUND");
totalCritical ++;
if (!fields.get(num).isNil()) {
filledCritical ++;
}
}
var answer = 0;
gs.print("TOTAL - " + totalCritical);
gs.print("FILLED - " + filledCritical);
if (filledCritical > 0 && totalCritical > 0) {
var pcnt = (filledCritical/totalCritical)*100;
answer = pcnt.toFixed(2);
}
answer;
```
Scoped GlideRecord - `getElement(String columnName)`

Retrieves the GlideElement object for the specified field.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnName</td>
<td>String</td>
<td>Name of the column to get the element from.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideElement</td>
<td>The GlideElement for the specified column of the current record.</td>
</tr>
</tbody>
</table>

```javascript
var elementName = 'short_description';
var gr = new GlideRecord('incident');
gr.initialize();
gr.setValue(elementName, "My DB is not working");
gr.insert();
gs.info(gr.getElement('short_description'));
```

Scoped GlideRecord - `getEncodedQuery()`

Retrieves the query condition of the current result set as an encoded query string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The encoded query as a string.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gr.addQuery('active', true);
gr.addQuery('priority', 1);
gr.query();
var encodedQuery = gr.getEncodedQuery();
gs.info(encodedQuery);
```

Output: `active=true^priority=1`
Scoped GlideRecord - `getLabel()`

Returns the field's label.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Field's label</td>
</tr>
</tbody>
</table>

```javascript
template.print("Summary of Requested items:\n");
var gr = new GlideRecord("sc_req_item");
gr.addQuery("request", current.sysapproval);
gr.query();
while(gr.next()) {
    var nicePrice = gr.price.toString();
    if (nicePrice != '') {
        nicePrice = parseFloat(nicePrice);
        nicePrice = nicePrice.toFixed(2);
    }
    template.print(gr.number + ":  " + gr.quantity + " X " +
        gr.cat_item.getDisplayValue() + " at $" + nicePrice + " each \n");
    template.print("    Options:\n");
    for (key in gr.variables) {
        var v = gr.variables[key];
        if(v.getGlideObject().getQuestion().getLabel() != '') {
            template.space(4);
            template.print('     ' + v.getGlideObject().getQuestion().getLabel() + " = 
                + v.getDisplayValue() + "\n");
        }
    }
}
```

Scoped GlideRecord - `getLastErrorMessage()`

Retrieves the last error message. If there is no last error message, null is returned.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The last error message as a string.</td>
</tr>
</tbody>
</table>

// Setup a data policy where short_description field in incident is mandatory
var gr = new GlideRecord('incident');
gr.insert(); // insert without data in mandatory field
var errormessage = gr.getLastErrorMessage();
gs.info(errormessage);

Output: Data Policy Exception: Short description is mandatory

Scoped GlideRecord - getLink(Boolean noStack)

Retrieves a link to the current record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>noStack</td>
<td>Boolean</td>
<td>If true, the sysparm_stack parameter is not appended to the link. The parameter sysparm_stack specifies the page to visit after closing the current link.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A link to the current record as a string.</td>
</tr>
</tbody>
</table>

gr = new GlideRecord('incident');
gr.addActiveQuery();
gr.addQuery("priority", 1);
gr.query();
gr.next();
gs.info(gs.getProperty('glide.servlet.uri') +
gr.getLink(false));

Output: <BaseURL>/incident.do?
sys_id=9d385017c611228701d22104cc95c371&sysparm_stack=incident_list.do?
sysparm_query=active=true

Scoped GlideRecord - getRecordClassName()

Retrieves the class name for the current record.

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The class name.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
var recordClassName = gr.getRecordClassName();
gs.info(recordClassName);
```

Output: incident

**Scoped GlideRecord - getRowCount()**

Retrieves the number of rows in the query result.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Number of rows.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gr.query();
gs.info("Records in incident table: " + gr.getRowCount());
```

**Scoped GlideRecord - getTableName()**

Retrieves the name of the table associated with the GlideRecord.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The table name</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gs.info(gr.getTableName());
```

 Scoped GlideRecord - getUniqueValue()

Gets the primary key of the record, which is usually the sys_id unless otherwise specified.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The unique primary key as a String, or null if the key is null.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('kb_knowledge');
gr.query();
gr.next();
var uniqueid = gr.getUniqueValue();
gs.info(uniqueid);
```

 Scoped GlideRecord - getValue(String name)

Retrieves the string value of an underlying element in a field.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the field to get the value from.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The string value of the underlying element. Returns null if the field is empty or the field does not exist. Boolean values return as &quot;0&quot; and &quot;1&quot; string values instead of false and true.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gr.orderBy('number');
gr.query('active','true');
gr.next();
gs.info(gr.getValue('number'));
```

Scoped GlideRecord - GlideRecord(String tableName)

Creates an instance of the GlideRecord class for the specified table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table to be used.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
```

Scoped GlideRecord - hasNext()

Determines if there are any more records in the GlideRecord object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if there are more records in the query result set.</td>
</tr>
</tbody>
</table>

```javascript
var rec = new GlideRecord('incident');
rec.query();
if (rec.hasNext()) {
  gs.info("Table is not empty");
```
Scoped GlideRecord - insert()

Inserts a new record using the field values that have been set for the current record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Unique ID of the inserted record, or null if the record is not inserted.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gr.initialize();
gr.name = 'New Incident';
gr.description = 'Incident description';
gr.insert();
```

Scoped GlideRecord - initialize()

Creates an empty record suitable for population before an insert.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var grIncident = new GlideRecord('incident');
grIncident.initialize();
grIncident.short_description='New Incident';
grIncident.description='Incident description';
grIncident.insert();
```

Scoped GlideRecord - isActionAborted()

Checks to see if the current database action is to be aborted.
isActionAborted() is initialized (set to false) for new threads and by the next() method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the current database action is to be aborted</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gs.info(gr.isActionAborted());
```

Output:

false

**Scoped GlideRecord - isNewRecord()**

Checks if the current record is a new record that has not yet been inserted into the database.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the record is new and has not been inserted into the database.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord("x_app_table");
gr.newRecord(); // create a new record and populate it with default values
gs.info(gr.isNewRecord());
```

**Scoped GlideRecord - isValid()**

Determines if the table exists.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the table is valid or if record was successfully retrieved. False if table is invalid or record was not successfully retrieved.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gs.info(gr.isValid());

var anotherGr = new GlideRecord('wrong_table_name');
gs.info(anotherGr.isValid());
```

Output: true false

Scoped GlideRecord - isValidField(String columnName)
Determines if the specified field is defined in the current table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnName</td>
<td>String</td>
<td>The name of the field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the field is defined for the current table.</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('incident');
gr.initialize();
gs.info(gr.isValidField("short_description"));
```

Scoped GlideRecord - isValidRecord()
Determines if a record was actually returned by the query/get record operation.
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Boolean     | Flag that indicates whether a record was actually returned by the query/get operation. Valid values:  
  - true: Record returned by query/get operation.  
  - false: End of record set, no record returned. |

```javascript
var rec = new GlideRecord('incident');
rec.query();
while (rec.next()) {
    gs.info(rec.number + ' exists');
}  
gs.info(rec.isValidRecord());
```

### Scoped GlideRecord - newRecord()

Creates a new GlideRecord record, sets the default values for the fields, and assigns a unique ID to the record.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord("x_app_table");
gr.newRecord();
gs.info(gr.isNewRecord());
```

Output: true

### Scoped GlideRecord - next()

Moves to the next record in the GlideRecord object.
**Note:** This method fails if there is a field in the table called "next". If that is the case, use the method `_next()`.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Flag that indicates if there is a &quot;next&quot; record in the GlideRecord.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>• true: Move to the next record was successful.</td>
</tr>
<tr>
<td></td>
<td>• false: No more records in the result set.</td>
</tr>
</tbody>
</table>

### Example

```javascript
var rec = new GlideRecord('incident');
rec.query();
while (rec.next()) {
    gs.info(rec.number + ' exists');
}
```

### Scoped GlideRecord - `_next()`

Moves to the next record in the GlideRecord. Provides the same functionality as `next()` , it is intended to be used in cases where the GlideRecord has a column named next.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if there are more records in the query set.</td>
</tr>
</tbody>
</table>
Scoped GlideRecord - operation()

Retrieves the current operation being performed, such as insert, update, or delete.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current operation.</td>
</tr>
</tbody>
</table>

//Commonly used in a business rule, returns insert if the current operation is insert
gs.info("current operation " + current.operation());

Scoped GlideRecord - orderBy(String name)

Specifies an orderBy column.

Call this method more than once to order by multiple columns. Results are arranged in ascending order, see orderByDesc(String name) to arrange records in descending order.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The column name used to order the records in this GlideRecord object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var queryString = "priority=2";
var gr = new GlideRecord('incident');
gr.orderBy('short_description'); // Ascending Order
gr.addEncodedQuery(queryString);
gr.query();
while (gr.next()) {
  gs.info(gr.short_description);
Scoped GlideRecord - orderByDesc(String name)

Specifies a decending orderBy column.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The column name to be used to order the records in a GlideRecord object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var queryString = "priority=2";
var gr = new GlideRecord('incident');
gr.orderByDesc('short_description'); //Descending Order
gr.addEncodedQuery(queryString);
gr.query();
while (gr.next()) {
    gs.info(gr.short_description);
}
```

Scoped GlideRecord - query(Object field, Object value)

Runs the query against the table based on the filters specified by addQuery, addEncodedQuery, etc.

This queries the GlideRecord table as well as any references of the table. Usually this is performed without arguments. If name/value pair is specified, "name=value" condition is added to the query.

Note: This method fails if there is a field in the table called "query". If that is the case, use the method _query().

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>Object</td>
<td>Column name to query on.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>Value to query for.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var rec = new GlideRecord('incident');
rec.query();
while (rec.next()) {
    gs.info(rec.number + ' exists');
}
```

Scoped GlideRecord - `_query(Object field, Object value)`

Identical to `query()`. This method is intended to be used on tables where there is a column named query, which would interfere with using the `query()` method.

Runs the query against the table based on the filters specified by the `addQuery()` and `addEncodedQuery()` methods. This method queries the GlideRecord table as well as any references of the table. Typically this method is called without arguments. If a name/value pair is specified, the “name=value” condition is added to the query.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Object</td>
<td>Column name on which to query</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>Value for which to query</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var rec = new GlideRecord('sys_app_module');
rec._query();
while (rec.next()) {
    gs.print(rec.number + ' exists');
}
```

Scoped GlideRecord - `setAbortAction(Boolean b)`

Sets a flag to indicate if the next database action (insert, update, delete) is to be aborted. This is often used in business rules.

Use in an onBefore business rule to prevent the database action from being done. The business rule continues to run after `setAbortAction()` is called. Calling `setAbortAction()` does not stop
subsequent business rules from executing. Calling this method only prevents the database action from occurring.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>True to abort the next action. False if the action is to be allowed.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
// Often used in business rule to check whether the current operation should be aborted.
if (current.size > 16) {
    current.setAbortAction(true);
}
```

**Scoped GlideRecord - setLimit(Number maxNumRecords)**

Sets the limit for number of records are fetched by the GlideRecord query.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxNumRecords</td>
<td>Number</td>
<td>The maximum number of records to fetch.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var gr = new GlideRecord('incident');
gr.orderByDesc('sys_created_on');
gr.setLimit(10);
gr.query(); // this retrieves latest 10 incident records created
```

**Scoped GlideRecord - setNewGuidValue(String guid)**

Sets sys_id value for the current record.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>guid</td>
<td>String</td>
<td>The GUID to be assigned to the current record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
ar gr = new GlideRecord('incident');
gr.short_description='The third floor printer is broken';
gr.setNewGuidValue('eb4636ca6f6d31005be8883e6b3ee333');
gr.insert();
gs.info(gr.sys_id);
```

Scoped GlideRecord - `setValue(String name, Object value)`

Sets the value of the field with the specified name to the specified value.

Normally the script does a `gr.category = value`. However, if the element name is itself a variable then `gr.setValue(elementName, value)` can be used. When setting a value, ensure the data type of the field matches the data type of the value you enter.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the field.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>The value to assign to the field.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var elementName = 'short_description';
var gr = new GlideRecord('incident');
gr.initialize();
gr.setValue(elementName, "My DB is not working");
gr.insert();
```

Scoped GlideRecord - `setWorkflow(Boolean enable)`

Enables or disables the running of business rules, script engines, and audit.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>Boolean</td>
<td>If true (default), enables business rules. If false, disables business rules.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
//Enable business rules, scripts engines for x_app_table
var gr = new GlideRecord("x_app_table");
gr.setWorkflow(true);
```

**Scoped GlideRecord - update(String reason)**

Updates the GlideRecord with any changes that have been made. If the record does not already exist, it is inserted.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reason</td>
<td>String</td>
<td>Optional. Reason for the update. The reason appears in the audit record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the new or updated record. Returns null if the update fails.</td>
</tr>
</tbody>
</table>

Example

```java
var gr = new GlideRecord('incident');
gr.get('99ebb4156fa831005be8883e6b3ee4b9');
gr.short_description='Update the short description';
gr.update();
gs.info(gr.getElement('short_description'));
```

**Scoped GlideRecord - updateMultiple()**

Updates each GlideRecord in a stated query with a specified set of changes.
When changing field values, use `setValue()` instead of directly setting the field (`field = something`). When using `updateMultiple()`, directly setting the field (`gr.state = 4`) results in all records in the table being updated instead of just the records returned by the query.

Do not use this method with the `chooseWindow()` or `setLimit()` methods when working with large tables.

This method sets new values and does not clear existing values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```javascript
// update the state of all active incidents to 4 - "Awaiting User Info"
var gr = new GlideRecord('incident');
gr.addQuery('active', true);
gr.setValue('state', 4);
gr.updateMultiple();
```

**GlideRecordUtil**

A utility class for working with GlideRecords.

The `GlideRecordUtil` class is available in server-side scripts.

**GlideRecordUtil - getCIGR(String sys_id)**

Returns a GlideRecord instance positioned to the given CI `sys_id`, and of the right class (table).

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>The <code>sys_id</code> of the desired CI.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>A GlideRecord instance positioned to the given CI <code>sys_id</code>, and of the right class (table).</td>
</tr>
</tbody>
</table>
```javascript
var gr = new GlideRecordUtil().getCIGR("2dfd7c8437201000deeabfc8bcbe5d56");

GlideRecordUtil - getFields(GlideRecord gr)

Returns a list of all the fields in the specified GlideRecord.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>A GlideRecord instance positioned to a valid record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>An array of field names for the specified GlideRecord.</td>
</tr>
</tbody>
</table>

```javascript
var queryString = "priority=1^ORpriority=2";
var gr = new GlideRecord('incident');
gr.addEncodedQuery(queryString);
gr.query();
gr.next();

var gRU = new GlideRecordUtil();
var fieldList = gRU.getFields(gr);
gs.info(fieldList);
```

Output: Line breaks added for presentation.

sys_id,skills,closed_by,assigned_to,contract,category,escalation,state,reassignment_count,time_worked,order,due_date,number,upon_approval,sys_tags,sla_due,follow_up,reopen_reason,caused_by,rejection_goto,assignment_group,comments_and_work_notes,incident_state,incident_number,incident_priority,incident_count,incident_category,incident_subcategory,incident_business_service,incident_wf_activity,incident_calendar_duration,incident_group_list,caller_id,comments,priority,sys_updated_by,variables,delivery_task,resolved_at,sys_updated_on,parent,active,opened_by,expected_start,work_notes,sys_created_by,additional_assignee_list,approval_set,cmdb_ci,user_input,sys_created_on,close_code,contact_type,resolved_by,rfc,approval_history,activity_due,severity,child_incidents,subcategory,work_end,closed_at,close_notes,variables,business_duration,knowledge,approval,sys_domain_path,sys_mod_count,problem_id,calendar_stc,work_start,sys_domain,correlation_id,sys_class_name,short_description,impact,description,correlation_display,urgency,made_sla,delivery_plan,work_notes_list

GlideRecordUtil - getGR(String baseTable, String sys_id)

Returns a GlideRecord instance for the given table, positioned to the given sys_id, and of the right class (table).

This method is useful when you need to load a GlideRecord from a sys_id, but you don't know what the actual table is (because it may be extended from the base table). This method always
returns a GlideRecord of the correct type base_table: the name of the base table that the specified sys_id is in.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseTable</td>
<td>String</td>
<td>The name of the base table containing the sys_id.</td>
</tr>
<tr>
<td>sys_id</td>
<td>String</td>
<td>The sys_id of the desired record.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The GlideRecord for the specified sys_id.</td>
</tr>
</tbody>
</table>

```java
var gr = new GlideRecordUtil().getGR("cmdb_ci_computer", "2dfd7c8437201000deeabfc8bcbe5d56");
```

**GlideRecordUtil - getTables(String tableName)**

Returns a Java ArrayList of the ancestors of the given table name.

For example, given cmdb_ci_linux_server, this would return cmdb_ci, cmdb_ci_computer, cmdb_ci_server, and cmdb_ci_linux_server.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Name of the table</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>A list of ancestors of the specified table</td>
</tr>
</tbody>
</table>

```java
var tables = new GlideRecordUtil().getTables("cmdb_ci_linux_server");
gs.log(tables.join(",")); // The result would be "cmdb_ci, cmdb_ci_computer, cmdb_ci_server, cmdb_ci_linux_server".
```

**GlideRecordUtil - mergeToGR(Object hashMap, GlideRecord gr, Object ignore)**

Sets the fields in the specified GlideRecord with the field values contained in the specified hashmap, unless that field name is in the ignore hashmap.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hashMap</td>
<td>Object</td>
<td>An Object instance (being used as a hashmap), with properties named for fields and containing the fields’ value.</td>
</tr>
<tr>
<td>GlideRecord</td>
<td>GR</td>
<td>The GlideRecord instance to receive the field values.</td>
</tr>
<tr>
<td>ignore</td>
<td>Object</td>
<td>An optional hashmap of field names to ignore.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

This example updates a computer record’s name and os fields, but does not update the sys_created_by field:

```javascript
var gr = new GlideRecordUtil().getGR("cmdb_ci_computer", "2dfd7c8437201000deeabfc8bcbe5d56");
var obj = {"name": "xyz", "os": "windows 2000", "sys_created_by": "aleck.lin"};
var ignore = {"sys_created_by": true};
new GlideRecordUtil().mergeToGR(obj, gr, ignore);
gr.update();
```

**GlideRecordUtil - populateFromGR(Object hashMap, GlideRecord gr, Object ignore)**

Populates the given hashmap from the given GlideRecord instance. Each field in the GlideRecord becomes a property in the hashmap.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hashMap</td>
<td>Object</td>
<td>An object being used as a hashmap.</td>
</tr>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>A GlideRecord instance positioned to a valid record.</td>
</tr>
<tr>
<td>ignore</td>
<td>Object</td>
<td>An optional hashmap of field names not to populate.</td>
</tr>
</tbody>
</table>
GlideRecordV3

GlideRecord is used for database operations. Client-side GlideRecord enables the use of some GlideRecord functionality in client-side scripts, such as client scripts and UI policy scripts.

A GlideRecord contains both records and fields.

Queries made with the client-side GlideRecord are executed on the server. Therefore, a request is made from the client browser to obtain the record data.

Client-side GlideRecord is not supported in scoped applications. Instead, create a script include and use GlideAjax, or use the REST APIs.

GlideRecordV3 - addOrderBy(String column)

Adds a column to order by in the query.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>String</td>
<td>The column by which to order the result set.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideRecordV3 - addQuery(String fieldName, Object value)

Adds a filter to return records where the field is equal to the value (or is in a list of values).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field to be checked.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>The value or list of values on which to query.</td>
</tr>
</tbody>
</table>

### GlideRecordV3 - addQuery(String fieldName, Object operator, Object value)

Adds a filter to return records where the field meets the specified condition (field, operator, value).

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field to be checked.</td>
</tr>
<tr>
<td>operator</td>
<td>Object</td>
<td>An operator for the query.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>The value to use.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### GlideRecordV3 - deleteRecord(Function responseFunction)

Deletes the current record.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>responseFunction</td>
<td>Function</td>
<td>The response function.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the record was deleted. False if no record was found to delete.</td>
</tr>
</tbody>
</table>

### GlideRecordV3 - get(String sysId)

Get a record by sysId.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysId</td>
<td>String</td>
<td>The sysID of the record for which to search.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if one or more matching records was found. False if no records were found.</td>
</tr>
</tbody>
</table>

#### GlideRecordV3 - getEncodedQuery()

Retrieves all query conditions as an encoded query string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>An encoded query string containing all conditions that have been added to the query.</td>
</tr>
</tbody>
</table>

#### GlideRecordV3 - getTableName()

Gets the name of the table associated with the GlideRecord.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The table name.</td>
</tr>
</tbody>
</table>

#### GlideRecordV3 - GlideRecord(String tableName)

Creates an instance of the GlideRecord class for the specified table.
var gr = new GlideRecord('incident');

**GlideRecordV3 - hasNext()**
Determines if there are any more records in the GlideRecord.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>String</td>
<td>The table to be used.</td>
</tr>
</tbody>
</table>

**GlideRecordV3 - insert(Function responseFunction)**
Inserts a new record using the field values that have been set for the current record.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>responseFunction</td>
<td>Function</td>
<td>The response function.</td>
</tr>
</tbody>
</table>

**GlideRecordV3 - next()**
Moves to the next record in the GlideRecord.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GlideRecordV3 - orderBy(String column)

Specifies an orderBy column. May be called more than once to order by multiple columns.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>String</td>
<td>The column to add to sort the result set.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideRecordV3 - query()

Performs a query. Takes zero or more parameters. Parameters may be in any order. Any function is considered to be a response function. Any pair of literals is considered a query pair (field : value). Do not make synchronous query calls. Performing a query without a response function makes the call synchronous, which means that the display will wait for the query response before continuing.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>responseFunction</td>
<td>Function</td>
<td>The function called when the query results are available. (optional)</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>A field name. (optional)</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The field value to check for. (optional)</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
GlideSchedule

The scoped GlideSchedule API provides methods for performing operations on GlideSchedule objects, such as adding new schedule segments to a schedule, determining if a datetime is within the schedule, or setting the schedule timezone.

Scoped GlideSchedule - add(GlideDateTime startDate, GlideDuration offset)

Adds a new schedule segment to the current schedule.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startDate</td>
<td>GlideDateTime</td>
<td>The starting date of the new schedule segment.</td>
</tr>
<tr>
<td>offset</td>
<td>GlideDuration</td>
<td>The time offset of the new schedule segment.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDateTime</td>
<td>The schedule updated with the new schedule segment.</td>
</tr>
</tbody>
</table>

```javascript
var startDate = new GlideDateTime('2014-01-02');
var days = 2;
var dur = new GlideDuration(60 * 60 * 24 * 1000 * days);
var schedule = new GlideSchedule();
var end = schedule.add(startDate, dur);
gs.info(end);
```

Output:

2014-01-04 00:00:00

Scoped GlideSchedule - duration(GlideDateTime startDate, GlideDateTime endDate)

Determines the elapsed time in the schedule between two date time values using the timezone of the schedule or, if that is not specified, the timezone of the session.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startDate</td>
<td>GlideDateTime</td>
<td>The starting datetime.</td>
</tr>
<tr>
<td>endDate</td>
<td>GlideDateTime</td>
<td>The ending datetime.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDuration</td>
<td>The difference between the starting and ending datetime.</td>
</tr>
</tbody>
</table>

```javascript
define('sys_id', '04e664654a36232701a2247dcd8fc4cf'); // sys_id for "Application" schedule record
var sched = new GlideSchedule(sys_id);
gs.info(sched.getName());```

### Scoped GlideSchedule - getName()

Retrieves the schedule name.

<table>
<thead>
<tr>
<th>Parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The name of the current schedule.</td>
</tr>
</tbody>
</table>

```javascript
define('sys_id', '04e664654a36232701a2247dcd8fc4cf'); // sys_id for "Application" schedule record
var sched = new GlideSchedule(sys_id);
gs.info(sched.getName());```

### Scoped GlideSchedule - GlideSchedule()

Instantiates an empty GlideSchedule object.
**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Scoped GlideSchedule - GlideSchedule(String sysID, String timeZone)**

Instantiates a GlideSchedule object and loads the schedule information. If a timezone is not specified or is nil, the current session timezone is used.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysID</td>
<td>String</td>
<td>The system ID for the schedule.</td>
</tr>
<tr>
<td>timeZone</td>
<td>String</td>
<td>The time zone. (Optional)</td>
</tr>
</tbody>
</table>

```javascript
var schedule = new GlideSchedule('090eeecae0a0a0b260077e1d7fda71da828', 'US/Pacific');
```

**Scoped GlideSchedule - isInSchedule(GlideDateTime time)**

Determines if the given datetime is within the current schedule.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>GlideDate</td>
<td>The datetime value to check.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the specified datetime is within the schedule; otherwise, false.</td>
</tr>
</tbody>
</table>

```javascript
var g = new GlideRecord('cmn_schedule');
g.addQuery('type', 'blackout');
g.query();
if (g.next()) {
  var sched = new GlideSchedule(g.sys_id);
  var d = new GlideDateTime();
  d.setDisplayValue("2007-09-18 12:00:00");
  if (sched.isInSchedule(d))
    gs.info("Is in the schedule");
  else
    gs.info("Is NOT in the schedule");
}
```
Scoped GlideSchedule - isValid()
Determines if the current schedule is valid. A schedule is valid if it has at least one schedule span.

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the schedule is valid.</td>
</tr>
</tbody>
</table>

```javascript
var g = new GlideRecord('cmn_schedule');
g.addQuery('type', 'blackout');
g.query();
if (g.next()) {
    var sched = new GlideSchedule(g.sys_id);
    var d = new GlideDateTime();
    dsetDisplayValue("2007-09-18 12:00:00");
    if (sched.isValid())
        gs.info("Is valid");
    else
        gs.info("Is not valid");
}
```

Scoped GlideSchedule - load(String sysID, String timeZone, String excludeSpanID)
Loads a schedule with the schedule information.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysID</td>
<td>String</td>
<td>The system ID of the schedule.</td>
</tr>
<tr>
<td>timeZone</td>
<td>String</td>
<td>(Optional) The timezone. If a timezone is not specified, or is nil, the current session timezone is used for the schedule.</td>
</tr>
<tr>
<td>excludeSpanID</td>
<td>String</td>
<td>Any span to exclude.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var x = new GlideSchedule();
x.load('08fcd0830a0a0b2600079f56b1adb9ae');
```

Scoped GlideSchedule - setTimeZone(String timeZone)
Sets the timezone for the current schedule.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeZone</td>
<td>String</td>
<td>The timezone.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

This example sets the timezone for the schedule to US/Pacific.

```javascript
var schedule = new GlideSchedule();
schedule.setTimeZone('US/Pacific');
```

Scoped GlideSchedule - `whenNext(GlideDateTime time, String timeZone)`

Determines how much time (in milliseconds) until start time of the next schedule item.

This function is intended to be called when the GlideSchedule object (cmn_schedule table) is not currently in the schedule window. The `whenNext()` call returns duration (in ms) until the GlideSchedule object is within the schedule. This function does not return a meaningful value if called when the GlideSchedule object is within the schedule.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>GlideDateTime</td>
<td>Time to be evaluated</td>
</tr>
<tr>
<td>timeZone</td>
<td>String</td>
<td>Timezone</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Number of milliseconds until the start time of the next schedule item. Returns -1 if never.</td>
</tr>
</tbody>
</table>

```javascript
var startDate = new GlideDateTime('2014-10-25 08:00:00');
var glideSchedule = new GlideSchedule('08fcd0830a0a0b2600079f56b1ad9ae', 'UTC');
gs.info(glideSchedule.whenNext(startDate));
```

Output:

172800000

testScript();
```javascript
function testScript() {
    var now = new GlideDateTime(); // current date and time
    var sched = new GlideSchedule("<sys_id>"); // Use a cmn_schedule
    if (sched.isInSchedule(now)) {
        gs.info('We are in an active schedule window so whenNext() is not helpful');
    } else {
        gs.info('Not currently in schedule so call whenNext()');
        var msUntilNext = sched.whenNext(new GlideDateTime(), 'US/Pacific');
        gs.info('Next schedule starts in ' + msUntilNext + ' milliseconds');
    }
}
```

Output:

(Scheduled for future) *** Script: Not currently in schedule *** Script: Next schedule starts in 332894000 milliseconds

**GlideScriptableInputStream**

A GlideScriptableInputStream object cannot be instantiated directly, but is used as an opaque object which is used to connect an input stream from GlideSysAttachment.getContentStream() with other streaming APIs, such as GlideTextReader, GlideDigest, and XMLDocument2.

See the scoped GlideSysAttachment API for methods that return a GlideScriptableInputStream object. The scoped GlideTextReader constructor requires a GlideScriptableInputStream object as an input parameter.

**GlideScriptedProcessor**

ServiceNow processors are equivalent to Java servlets.

Processors provide a customizable URL endpoint that can execute arbitrary server-side JavaScript code and produce output such as TEXT, JSON, or HTML. The GlideScriptedProcessor APIs are used in processor scripts to access the processor (servlet) capabilities. There are no constructors for the GlideScriptedProcessor APIs. The methods are called using the global variable `g_processor`.

A useful global variable, `g_target`, is available in processor scripts. It contains the table name extracted from the URL.

The URL to a processor has the format: `https://<instance name.servicenow.com>/<path endpoint>.do?<parameter endpoint>=<value>` where the path endpoint and parameter endpoint are defined on the processor form.

**Scoped GlideScriptedProcessor - redirect(String url)**

Redirects to the specified URL.

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>String</td>
<td>the destination URL</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

//Do whatever processing you need and redirect to the homepage

g_processor.redirect("/navpage.do")

Scoped GlideScriptedProcessor - writeJSON(Object o)

Encodes an object as a JSON string and writes it to the current URL.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Object</td>
<td>The object to encode to a JSON string.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var map = {"key1":"value1","key2":"value2"};
g_processor.writeJSON(map);

Scoped GlideScriptedProcessor - writeOutput(String s)

Writes the specified string to the current URL.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>String</td>
<td>The string to write.</td>
</tr>
</tbody>
</table>
var name = g_request.getParameter("name");
g_processor.writeOutput("Hello "+ name);

**Scoped GlideScriptedProcessor - writeOutput(String contentType, String s)**

Writes the specified string to the current URL in the specified character-encoding.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contentType</td>
<td>String</td>
<td>Sets the content type of the response sent to the client, if the response has not been committed, and may include a character-encoding specification.</td>
</tr>
<tr>
<td>s</td>
<td>String</td>
<td>The string to write.</td>
</tr>
</tbody>
</table>

**GlideSecureRandomUtil**

The scoped GlideSecureRandomUtil API provides methods for generating integers, long values, and strings.

There is no constructor for this class. Methods are accessed through the static object GlideSecureRandomUtil. The GlideSecureRandomUtil class is available in both global and scoped applications.

**Scoped GlideSecureRandomUtil - getSecureRandomInt()**

Generates a pseudo-random integer.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The pseudo-randomly generated integer.</td>
</tr>
</tbody>
</table>

```java
gs.info(GlideSecureRandomUtil.getSecureRandomInt());
```

Output:

```
1976146969
```

Scoped GlideSecureRandomUtil - getSecureRandomIntBound(Number bound)

Generates a pseudo-random integer between 0 (inclusive) and the bound (exclusive) value that you pass into the method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bound</td>
<td>Number</td>
<td>The bound value.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The pseudo-randomly generated integer.</td>
</tr>
</tbody>
</table>

```java
gs.info(GlideSecureRandomUtil.getSecureRandomIntBound(100));
```

Output:

```
55
```

Scoped GlideSecureRandomUtil - getSecureRandomLong()

Generates pseudo-random long value.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The pseudo-randomly generated 64-bit integer.</td>
</tr>
</tbody>
</table>

```java
gs.info(GlideSecureRandomUtil.getSecureRandomLong());
```

Output:

```
792836514424092500
```

## Scoped GlideSecureRandomUtil - getSecureRandomString(Number length)

Generates a random alpha-numeric string with the specified length.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>Number</td>
<td>The length of the string in number of characters.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The randomly generated string.</td>
</tr>
</tbody>
</table>

```java
gs.info(GlideSecureRandomUtil.getSecureRandomString(12));
```

Output:

```
1XzVI0sLfV1T
```

## GlideServletResponse

The GlideServletRequest API is used in processor scripts.

ServiceNow processors are equivalent to Java servlets. Processors provide a customizable URL endpoint that can execute arbitrary server-side JavaScript code and produce output such as TEXT, JSON, or HTML. The GlideServletRequest API is used in processor scripts to access
the `HttpServletRequest` object. The `GlideServletRequest` object provides a subset of the `HttpServletRequest` APIs. The methods are called using the global variable `g_request`.

A useful global variable, `g_target`, is available in processor scripts. It contains the table name extracted from the URL.

The URL to a processor has the format: `https://<instance name.service-now.com>/<path endpoint>.do?<parameter endpoint>=<value>` where the path endpoint and parameter endpoint are defined on the processor form.

**Scoped GlideServletRequest - `getContentType()`**

Returns the MIME type of the body of the request.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The content type, returns null if the content type is not known.</td>
</tr>
</tbody>
</table>

```javascript
var contentType = g_request.getContentType();
```

**Scoped GlideServletRequest - `getHeader(String name)`**

Returns the header value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the header to be retrieved.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The header.</td>
</tr>
</tbody>
</table>

```javascript
var headerValue = g_request.getHeader("host");
```

Output:

demonightlyus.service-now.com
**Scoped GlideServletRequest - getHeaderNames()**

Returns a comma-separated list of header names.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A comma-separated list of header names.</td>
</tr>
</tbody>
</table>

```javascript
var headerList = g_request.getHeaderNames();
```

Output:

```javascript
host,connection,cache-control,authorization,accept,user-agent,accept-encoding,accept-language,cookie,x-forwarded-proto,x-forwarded-host,x-forwarded-for
```

**Scoped GlideServletRequest - getHeaders(String name)**

Returns the header values.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Names of the headers to be retrieved.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The header values.</td>
</tr>
</tbody>
</table>

```javascript
var headerValue = g_request.getHeaders("host");
```

Output:

```javascript
demonightlyus.service-now.com
```

**Scoped GlideServletRequest - getParameter(String name)**

Returns the value of the parameter contained in the request URL.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the parameter to be retrieved. This can be the parameter endpoint from the processor form.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strings</td>
<td>The parameter value. Returns null if the parameter is not found.</td>
</tr>
</tbody>
</table>

```javascript
var name = g_request.getParameter("x_snc_custom_x_snc_name");
```

### Scoped GlideServletRequest - getParameterNames()

Returns an enumeration list of URL parameters that were used in the request URI.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration</td>
<td>Java enumerator object list of URL parameters used in the HTTP request URI.</td>
</tr>
</tbody>
</table>

```javascript
var paramList = g_request.getParameterNames();
```

### Scoped GlideServletRequest - getQueryString()

Returns the query string from the request.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
```
The code snippet is:

```javascript
var daString = g_request.getQueryString();
g_processor.writeOutput("The query string is: " + daString);
```

Output: The request URL is: https://demonightlycloudedge.service-now.com/x_custom_app_customApp.do?x_custom_app_name=George&bell=rung

The query string is: x_custom_app_name=George&bell=rung

### GlideServletResponse

The GlideServletResponse API is used in processor scripts.

ServiceNow processors are equivalent to Java servlets. Processors provide a customizable URL endpoint that can execute arbitrary server-side JavaScript code and produce output such as TEXT, JSON, or HTML. The GlideServletResponse API is used in processor scripts to access the HttpServletResponse object. The GlideServletResponse object provides a subset of the HttpServletResponse APIs. The methods are called using the global variable g_response.

A useful global variable, g_target, is available in processor scripts. It contains the table name extracted from the URL.

The URL to a processor has the format: https://<instance name.servicenow.com>/<path endpoint>.do?<parameter endpoint>=<value> where the path endpoint and parameter endpoint are defined on the processor form.

### Scoped GlideServletResponse - sendRedirect(String location)

Sends a temporary redirect to the client.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>String</td>
<td>The URL to receive the response.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### Scoped GlideServletResponse - setContentType(String type)

Sets the MIME type of the response
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>The MIME type.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
g_response.setContentType('text/html;charset=UTF-8');
```

#### Scoped GlideServletResponse - setHeader(String key, String value)

Sets a response header to the specified value.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>String</td>
<td>Specifies the header.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to be assigned to the header. If the header exists, it is overwritten.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
g_response.setHeader("host", "demonightlyus.service-now.com");
```

#### Scoped GlideServletResponse - setStatus(Number status)

Sets the status code for the response.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Number</td>
<td>The status to be set.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
// set the status to okay
g_response.setStatus(200);

### GlideSession

The GlideSession API allows you to find information about the current session.

### GlideSession - clearClientData(String paramName)

Clears a session client value previously set with `putClientData()`.

This method is used in a client script to clear data values that were set by a server script using the `putClientData()` method.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramName</td>
<td>String</td>
<td>Name of the client data to clear.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
session.putClientData('custName', 'Harry');
var clientData = session.getClientData('custName');
gs.info(clientData);
session.clearClientData('custName');
clientData = session.getClientData('custName');
gs.info(clientData);
```

Output:

Harry
null

### GlideSession - getClientData(String paramName)

Returns a session client value previously set with `putClientData()`.
This method is used in a client script to retrieve data values that were set by a server script that used the `putClientData()` method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramName</td>
<td>String</td>
<td>Name of the client data to retrieve.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The client data as a string.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
session.putClientData('test1', 'Harry');
var clientData = session.getClientData('test1');
gs.info(clientData);
```

**Output:**

Harry

**Scoped equivalent**

To use the `getClientData()` method in a scoped application, use the corresponding scoped method: `getClientData()`.

**GlideSession - getLanguage()**

Returns the session's language code.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The session's language code.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
var language = session.getLanguage();
gs.info(language);
```
Scoped equivalent

To use the `getLanguage()` method in a scoped application, use the corresponding scoped method: `getLanguage()`.

**GlideSession - getRoles()**

Returns a list of roles for the current user.

The list of roles does not reflect any changes made during the current user session. To get the updated list of roles, the user must log out and log back in.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A comma separated list of roles.</td>
</tr>
</tbody>
</table>

```java
gs.info(gs.getSession().getRoles());
```

**Output:**

```
admin,hr_fulfiller,itsa_fulfiller,security_admin
```

**GlideSession - getTimeZoneName()**

Returns the name of the session's time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The name of the session's time zone.</td>
</tr>
</tbody>
</table>
var session = gs.getSession();
var zoneName = session.getTimeZoneName();
gs.info(zoneName);

Output:

US/Pacific

Scoped equivalent

To use the `getTimeZoneName()` method in a scoped application, use the corresponding scoped method: `getTimeZoneName()`.

GlideSession - isInteractive()

Determines if the current session is interactive.

An interactive session is one that involves an end-user interacting with a user interface that then retrieves information from a server. An example of this type of session is when a user logs in using the log-in screen or uses a form to query a data store. A non-interactive session is one that only involves programmatic interaction with a server such as a SOAP request to retrieve data.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the session is interactive.</td>
</tr>
</tbody>
</table>

var interActive = gs.getSession().isInteractive();
gs.info(interActive);

Output:

false

GlideSession - isLoggedIn()

Determines if the current user is currently logged in.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the current user is logged in.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
var loggedIn = session.isLoggedIn();
gs.info(loggedIn);
```

Output:

```
true
```

Scoped equivalent

To use the `isLoggedIn()` method in a scoped application, use the corresponding scoped method: `isLoggedIn()`.

**GlideSession - putClientData(String paramName, String paramValue)**

Sets a session client value that can be retrieved with `getClientData()`. This method is used in a server side script that runs when a form is created.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramName</td>
<td>String</td>
<td>Name of the client parameter to set.</td>
</tr>
<tr>
<td>paramValue</td>
<td>String</td>
<td>Parameter value.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
session.putClientData('test1', 'Harry');
var clientData = session.getClientData('test1');
```
Scoped equivalent

To use the `putClientData()` method in a scoped application, use the corresponding scoped method: `putClientData()`.  

GlideSession

The scoped GlideSession API provides a way to find information about the current session. There are no constructors for creating an instance of a scoped GlideSession object. Instead, use the `getSession()` method of the scoped GlideSystem API.

Scoped GlideSession - getClientData(String paramName)

Returns a session client value previously set with `putClientData()`.

This method is used in a client script to retrieve data values that were set by a server script that used the `putClientData()` method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramName</td>
<td>String</td>
<td>Name of the client data to retrieve.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The client data as a string.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
session.putClientData('test1', 'Harry');
var clientData = session.getClientData('test1');
gs.info(clientData);
```

Output:

Harry

Scoped GlideSession - getClientIP()

Returns the client IP address.
**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The IP address.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
var addr = session.getClientIP();
gs.info(addr);
```

Output:

50.59.164.97

**Scoped GlideSession - getCurrentApplicationId()**

Returns the application currently selected in the application picker.

This method requires admin privileges.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The currently selected application.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
var appID = session.getCurrentApplicationId();
gs.info(appID);
```

Output:

ce05b9f32b840200c5244f74b4da1501

**Scoped GlideSession - getLanguage()**

Returns the session’s language code.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The session's language code.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
var language = session.getLanguage();
gs.info(language);
```

Output:
```
en
```

**Scoped GlideSession - getSessionToken()**

Returns the session token.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The session token.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
var token = session.getSessionToken();
gs.info(token);
```

Output:
```
4284b5372b840200c5244f74b4da15f2c3476cf7fcb6572afa4ef9d5e6d307a5fd9e1da7
```

**Scoped GlideSession - getTimeZoneName()**

Returns the name of the session's time zone.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The name of the session's time zone.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
var zoneName = session.getTimeZoneName();
gs.info(zoneName);
```

Output:

US/Pacific

**Scoped GlideSession - getUrlOnStack()**

Returns the URL on the stack. Returns null if the stack is empty.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The URL. Returns null if the stack is empty.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
var URL = session.getUrlOnStack();
gs.info(URL);
```

Output: line breaks added for clarity.

```
sys_app.do?
sys_id=ce05b9f32b840200c5244f74b4da1501&sysparm_goto_url=sys_app.do %3Fsys_id%3Dce05b9f32b840200c5244f74b4da1501
```

**Scoped GlideSession - isImpersonating()**

Returns true if the user is impersonating another user.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the user is impersonating another user; otherwise, returns false.</td>
</tr>
</tbody>
</table>

```javascript
var isImpersonator = gs.getSession().isImpersonating();
gs.info(isImpersonator);
```

**Scoped GlideSession - isInteractive()**

Returns true if the session is interactive.

An interactive session is one that involves an end-user interacting with a user interface that then retrieves information from a server. An example of this type of session is when a user logs in using the log-in screen or uses a form to query a data store. A non-interactive session is one that only involves programmatic interaction with a server such as a SOAP request to retrieve data.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the session is interactive.</td>
</tr>
</tbody>
</table>

```javascript
var interActive = gs.getSession().isInteractive();
gs.info(interActive);
```

**Scoped GlideSession - isLoggedIn()**

Returns true if the user is logged in.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var isLoggedIn = gs.getSession().isLoggedIn();
gs.info(isLoggedIn);
```
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user is logged in.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
var loggedIn = session.isLoggedIn();
gs.info(loggedIn);
```

Output:

```
true
```

Scoped GlideSession - putClientData(String paramName, String paramValue)

Sets a session client value that can be retrieved with `getClientData()`. This method is used in a server side script that runs when a form is created.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramName</td>
<td>String</td>
<td>Name of the client data to set.</td>
</tr>
<tr>
<td>paramValue</td>
<td>String</td>
<td>Value of the client data.</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
session.putClientData('test1', 'Harry');
var clientData = session.getClientData('test1');
gs.info(clientData);
```

Output:

```
Harry
```

GlideSPScriptable

The GlideSPScriptable API provides a set of methods for use in Service Portal Widgets. You access GlideSPScriptable methods by using the global `$sp` object.

**Scoped GlideSPScriptable - canReadRecord(GlideRecord gr)**

Returns true if the user can read the specified GlideRecord.
If the record type is `kb_knowledge`, `sc_cat_item`, or `sc_category`, the method checks if the user can view the item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>The GlideRecord to check.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the record is valid and readable.</td>
</tr>
</tbody>
</table>

**Server script**

```javascript
//Server script
data.items = [];
data.userName = gs.getUserDisplayName();
var gr = new GlideRecord("sc_cat_item");
gr.query();
while (gr.next() && data.items.length < 10) {
    if ($sp.canReadRecord(gr)) {
        data.items.push(gr.getDisplayValue("name"));
    }
}

//HTML template
<div class="panel panel-default">
    <div class="panel-heading">Hi, {{c.data.userName}}!</div>
    <div class="panel-body">
        Here are some things you can order:
        <ul>
            <li ng-repeat="item in c.data.items">{{item}}</li>
        </ul>
    </div>
</div>
```

**Scoped GlideSPScriptable - canReadRecord(String table, String sysId)**

Returns true if the user can read the specified GlideRecord.

If the record type is `kb_knowledge`, `sc_cat_item`, or `sc_category`, the method checks if the user can view the item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>Name of the table to query.</td>
</tr>
<tr>
<td>sysId</td>
<td>String</td>
<td>Sys_id of the record to query.</td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the record is valid and readable.</td>
</tr>
</tbody>
</table>

### Scoped GlideSPScriptable - canSeePage(String pageID)

Returns true if the currently logged in user has permission to view the specified page.

The system determines permission using roles and user criteria. For more information, see [Configure page security by role](configure_page_security_by_role.html) and [User criteria for Service Portal](user_criteria_for_service_portal.html).

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageID</td>
<td>String</td>
<td>Page ID from the Pages (sp_page) table.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Boolean| Flag that indicates whether the currently logged in user has permissions to view the specified page. Values include:  
  - true: The user can view the page.  
  - false: Either the user cannot view the page, the given pageID is not valid, or the user is not logged in. |

This example adds a condition to the Knowledge menu item in the SP Header Menu to only show the menu item to users who have permission to view the kb_view2 page.

```javascript
GlideSPScriptable.canSeePage("kb_view2")
```

### Scoped GlideSPScriptable - getCatalogItem(String sysId)

Returns a model and view model for a `sc_cat_item` or `sc_cat_item_guide`.

This method is deprecated. Use the `getCatalogItem(String sysId, Boolean isOrdering)` method instead. This method calls the `getCatalogItem(String sysId, Boolean isOrdering)` method with the `isOrdering` parameter set to false, which means that write roles security checking is done.

This method is a quick way to get the data necessary to render and order a catalog item using `<sp-model />`. If you just need to get a catalog item to show its picture or name, use GlideRecord to query the `sc_cat_item` table.
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysId</td>
<td>String</td>
<td>The sys_id of the catalog item (sc_cat_item) or order guide (sc_cat_item_guide).</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object containing the catalog item variable model, view, sections, pricing, and client scripts.</td>
</tr>
</tbody>
</table>

```javascript
// Server script
(function() {
  var sys_id = $sp.getParameter("sys_id")
  data.catItem = $sp.getCatalogItem(sys_id);
})();

// Client script
function($http, spUtil) {
  var c = this;
  var submitting = false;
  c.getIt = function() {
    if (submitting) return;
    $http.post(spUtil.getURL('sc_cat_item'),
    c.data.catItem).success(function(response) {
      if (response.answer) {
        c.req = response.answer;
        c.req.page = c.req.table == 'sc_request' ?
' sc_request' : 'ticket';
      }
    });
  }
}

//SCSS
.img-bg {
  padding: 5px;
  background-color: $brand-primary;
}
.img-responsive {
  margin: 0 auto;
}
.cat-icon {
  display: block;
  margin: -40px auto 0;
}

// HTML template
<div class="col-sm-4">
  <div class="panel panel-default">
    <div class="img-bg">
      <img ng-src="{{::data.catItem.picture}}" class="img-responsive" />
    </div>
  </div>
</div>
```
Scoped GlideSPScriptable - getCatalogItem(String sysId, Boolean isOrdering)

Returns a model and view model for a `sc_cat_item` or `sc_cat_item_guide`.

See [Apply permissions to a service catalog variable](#) for information on security roles for Service Catalog variables.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysId</td>
<td>String</td>
<td>The sys_id of the catalog item (<code>sc_cat_item</code>) or order guide (<code>sc_cat_item_guide</code>).</td>
</tr>
<tr>
<td>isOrdering</td>
<td>Boolean</td>
<td>When true, uses create roles security check. When false, uses write roles security check. When users are ordering an item or have it in their cart, check using the create roles. If users are not ordering, for example, somebody is looking at a requested item to see the variables associated with that item, then check using the write roles.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object containing the catalog item variable model, view, sections, pricing, and client scripts.</td>
</tr>
</tbody>
</table>

Scoped GlideSPScriptable - getDisplayValue(String fieldName)

Returns the display value of the specified field (if it exists and has a value) from either the widget's sp_instance or the sp_portal record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The display value from either the sp_instance or sp_portal record.</td>
</tr>
</tbody>
</table>

//Server script
(function() {
    data.title = $sp.getDisplayValue("title");
    data.catalog = $sp.getDisplayValue("sc_catalog");
})();

//HTML template
<div>
    <h1>sp_instance.title: {{::data.title}}</h1>
    <h1>sp_portal.sc_catalog: {{::data.catalog}}</h1>
</div>

Scoped GlideSPScriptable - getField(GlideRecord gr, String fieldName)

Returns information about the specified field in the specified GlideRecord.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>The GlideRecord to check</td>
</tr>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field to find information for</td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object containing the field's label, value, displayValue, and type. Returns null if the GlideRecord of field name are not valid, or if the field is not readable.</td>
</tr>
</tbody>
</table>

**Scoped GlideSPScriptable - getFields(GlideRecord gr, String fieldNames)**

Checks the specified list of field names, and returns an array of valid field names.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>The GlideRecord to check</td>
</tr>
<tr>
<td>field Names</td>
<td>String</td>
<td>A comma separated list of field names.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of valid fields.</td>
</tr>
</tbody>
</table>

**Scoped GlideSPScriptable - getFieldsObject(GlideRecord gr, String fieldNames)**

Checks the specified list of field names and returns an object of valid field names.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>The GlideRecord to check</td>
</tr>
<tr>
<td>field Names</td>
<td>String</td>
<td>A comma separated list of field names.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object containing valid field names.</td>
</tr>
</tbody>
</table>

**Scoped GlideSPScriptable - getForm(String tableName, String sysId)**

Return the form.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The name of the table</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>sysId</td>
<td>String</td>
<td>The form's sys_id</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The form</td>
</tr>
</tbody>
</table>

**Scoped GlideSPScriptable - getKBCategoryArticles(String sys_id, Number limit)**

Returns KB articles in the specified category and its subcategories.

To avoid performance issues, do not use this method to return articles in large categories or articles with inline images. Instead, use `getKBArticleSummaries()`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>Sys_id of the KB article category.</td>
</tr>
<tr>
<td>limit</td>
<td>Number</td>
<td>Maximum number of KB articles returned.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>The articles within the category and its subcategories with:</td>
</tr>
<tr>
<td></td>
<td>• A workflow_state of published.</td>
</tr>
<tr>
<td></td>
<td>• A valid_to date greater than or equal to the current date.</td>
</tr>
</tbody>
</table>

```javascript
//Server script
(function() {
  data.kbs = $sp.getKBCategoryArticles("0ac1bf8bff0221009b20ffffffffffec", 5);
})();

//HTML template
<div>
  articles: {{::data.kbs}}
</div>
```

**Scoped GlideSPScriptable - getKBCategoryArticleSummaries(String sys_id, Number limit, Number maxChars)**

Returns Knowledge Base article summaries in the specified category and its subcategories.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>Sys_id of the KB article category.</td>
</tr>
<tr>
<td>limit</td>
<td>Number</td>
<td>Maximum number of KB articles returned.</td>
</tr>
<tr>
<td>maxChars</td>
<td>Number</td>
<td>Maximum number of characters to return from the article text. For full article text, set the value to -1.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>The articles within the category and its subcategories with:</td>
</tr>
<tr>
<td></td>
<td>• A workflow_state of published.</td>
</tr>
<tr>
<td></td>
<td>• A valid_to date greater than or equal to the current date.</td>
</tr>
</tbody>
</table>

//Server script
(function() {
  data.summary = $sp.getKBCategoryArticleSummaries("0ac1bf8bff0221009b20ffffffffffec", 5, 200);
})();

//HTML template
<div>
  articles: {{::data.summary}}
</div>

Scoped GlideSPScriptable - getKBCount(String sys_id)

Returns the number of articles in the defined Knowledge Base.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>Sys_id of a Knowledge Base record.</td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Number of knowledge articles in the defined Knowledge Base with:</td>
</tr>
<tr>
<td></td>
<td>• A workflow_state of published.</td>
</tr>
<tr>
<td></td>
<td>• A valid_to date greater than or equal to the current date.</td>
</tr>
</tbody>
</table>

---

### Server script
```javascript
(function() {
    data.count = $sp.getKBCount("a7e8a78bfff0221009b20fffffffff17");
})();
```

### HTML template
```html
<div>
    articles: {{::data.count}}
</div>
```

## Scoped GlideSPScriptable - `getListColumns(String tableName, String view)`

Returns a list of the specified table's columns in the specified view.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Name of the table</td>
</tr>
<tr>
<td>view</td>
<td>String</td>
<td>The view by which to filter the columns</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object containing the column names.</td>
</tr>
</tbody>
</table>

## Scoped GlideSPScriptable - `getMenuItems(String sysId)`

Returns an array of menu items for the specified instance.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysId</td>
<td>String</td>
<td>sysId of the instance</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Array</td>
<td>Menu items for the specified instance</td>
<td></td>
</tr>
</tbody>
</table>

**Scoped GlideSPScriptable - getMenuHREF(GlideRecord page)**

Returns the (?id=) portion of the URL based on the sp_menu type.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>GlideRecord</td>
<td>The page</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The href portion of the URL.</td>
</tr>
</tbody>
</table>

**Scoped GlideSPScriptable - getParameter(String name)**

Returns the value of the specified parameter.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the key from the query string or post body.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Returns the specified parameter as an object. Returns null if there is no request, JSON request, or widget.</td>
</tr>
</tbody>
</table>

**Scoped GlideSPScriptable - getPortalRecord()**

Returns the portal's GlideRecord.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The portal record</td>
</tr>
</tbody>
</table>

```javascript
//Server script
(function() {
  var portalGr = $sp.getPortalRecord();
  data.logo = portalGr.getDisplayValue("logo");
  data.homepage = portalGr.getDisplayValue("homepage.id");
})();
```

```html
//HTML template
<div>
  <img ng-src="{{::c.data.logo}}" />
  <a href="?id={{::c.data.homepage}}">Click here to go home</a>
</div>
```

### Scoped GlideSPScriptable - getRecord(String table, String sys_id)

If parameters are provided, returns the GlideRecord identified by the provided table and Sys ID. If no parameters are provided, returns the record identified by the current URL.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>Optional. The table of the record to return. If no parameters are included, returns the table and Sys ID identified by the current URL.</td>
</tr>
<tr>
<td>sys_id</td>
<td>String</td>
<td>Optional. The Sys ID of the record to return. If no parameters are included, returns the record identified by the current URL.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>If parameters are provided, returns the record identified by the provided table and Sys ID. If no parameters are provided, returns the record identified by the current URL. Returns null if the widget is embedded by another widget, or if the record for the provided parameters is not found.</td>
</tr>
</tbody>
</table>

```javascript
(function(){
  var gr = $sp.getRecord();
  data.tableLabel = gr.getLabel();
})();

// HTML template
<div class="panel-heading">
  <h4 class="panel-title">${{{data.tableLabel}}} details</h4>
</div>
```

Scoped GlideSPScriptable - getRecordDisplayValues(Object data, GlideRecord from, String names)

Copies display values for the specified fields into the data parameter.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Object</td>
<td>The display values for the specified fields are copied to this object.</td>
</tr>
<tr>
<td>from</td>
<td>GlideRecord</td>
<td>The GlideRecord to process.</td>
</tr>
<tr>
<td>names</td>
<td>String</td>
<td>A comma-separated list of field names.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Scoped GlideSPScriptable - getRecordElements(Object data, GlideRecord from, String names)

For the specified fields, copies the element's name, display value, and value into the data parameter.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Object</td>
<td>The element's name, display value, and value for the specified fields are copied to this object.</td>
</tr>
<tr>
<td>from</td>
<td>GlideRecord</td>
<td>The GlideRecord to process.</td>
</tr>
<tr>
<td>names</td>
<td>String</td>
<td>A comma-separated list of field names.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Scoped GlideSPScriptable - getRecordValues(Object data, GlideRecord from, String names)

Copies values for the specified field names from the GlideRecord into the data parameter.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Object</td>
<td>The value for the specified fields are copied to this object.</td>
</tr>
<tr>
<td>from</td>
<td>GlideRecord</td>
<td>The GlideRecord to process.</td>
</tr>
<tr>
<td>names</td>
<td>String</td>
<td>A comma-separated list of field names.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Scoped GlideSPScriptable - getRecordVariables(GlideRecord gr, Boolean includeNilResponses)

Returns Service Catalog variables associated with a record in String format.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>The record to retrieve Service Catalog variables for. Must be a record with Service Catalog variables defined, such as a requested item (sc_req_item) record or an incident submitted through a record producer.</td>
</tr>
<tr>
<td>includeNilResponses</td>
<td>Boolean</td>
<td>Optional. If true, the API includes variables with no user-defined value.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Service Catalog variables associated with the record.</td>
</tr>
</tbody>
</table>

//Server script
(function() {
    var itemsGR = new GlideRecord("sc_req_item");
    itemsGR.get('585d1bc4f4f13008a959a211310c77d');

    data.scVars = $sp.getRecordVariables(itemsGR);
})();

//HTML template
<div>
    Requested item variables: {{::data.scVars}}
</div>

Output:

Requested item variables: [
    {
        "display_value":"MetroPCS",
        "name":"carrier",
        "visible_summary":true,
        "id":"585d1bc44f4f13008a959a211310c77c",
        "label":"Allocated carrier",
        "type":5,
        "value":"metropcs",
        "visible_standalone":true,
        "visible_guide":true
    },
    {
        "display_value":"Unlimited",
        "name":"data_plan",
        "visible_summary":true,
        "id":"d05d1bc44f4f13008a959a211310c77c",
        "label":"Monthly data allowance",
        "type":5,
    },
Scoped GlideSPScriptable - getRecordVariablesArray(GlideRecord gr, Boolean includeNilResponses)

Returns an array of Service Catalog variables associated with a record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>The record to retrieve Service Catalog variables for. Must be a record with Service Catalog variables defined, such as a requested item (sc_req_item) record or an incident submitted through a record producer.</td>
</tr>
<tr>
<td>includeNilResponses</td>
<td>Boolean</td>
<td>Optional. If true, the API includes variables with no user-defined value.</td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Array of Service Catalog variables associated with the record.</td>
</tr>
</tbody>
</table>

---

```javascript
//Server script
(function() {
  var itemsGR = new GlideRecord("sc_req_item");
  itemsGR.get('585d1bc44f4f13008a959a211310c77d');
  data.scVars = $sp.getRecordVariablesArray(itemsGR);
})();

//HTML template
<div>
  Requested item variables: {{::data.scVars}}
</div>
```

**Output:**

Requested item variables: [  
  {  
    "display_value":"MetroPCS",  
    "name":"carrier",  
    "visible_summary":true,  
    "id":"585d1bc44f4f13008a959a211310c77c",  
    "label":"Allocated carrier",  
    "type":5,  
    "value":"metropcs",  
    "visible_standalone":true,  
    "visible_guide":true  
  },  
  {  
    "display_value":"Unlimited",  
    "name":"data_plan",  
    "visible_summary":true,  
    "id":"d05d1bc44f4f13008a959a211310c77c",  
    "label":"Monthly data allowance",  
    "type":5,  
    "value":"unlimited",  
    "visible_standalone":true,  
    "visible_guide":true  
  },  
  {  
    "display_value":"12 Months",  
    "name":"duration",  
    "visible_summary":true,  
    "id":"d85d1bc44f4f13008a959a211310c77c",  
    "label":"Contract duration",  
    "type":5,  
    "value":"twelve_months",  
    "visible_standalone":true,  
    "visible_guide":true  
  },  
  {  
    "display_value":"Slate",  
    "name":"color",  
```
Scoped GlideSPScriptable - getStream(String table, String sysId)

Gets the activity stream for the specified record. This method works on tables that extend the task table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>The table name</td>
</tr>
<tr>
<td>sysID</td>
<td>String</td>
<td>The sys_id of the record</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>If a table extending the task table is specified, contains the display_value, sys_id, short_description, number, entries, user_sys_id, user_full_name, user_login, label, table, and journal_fields properties; otherwise contains the table and sys_id properties.</td>
</tr>
</tbody>
</table>

*Note:* The user_login property contains the User ID of the current user. The user_sys_id and user_full_name properties reference the creator of the queried record.

Scoped GlideSPScriptable - getUserInitials()

Returns the user's initials.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The user's initials</td>
</tr>
</tbody>
</table>

**Scoped GlideSPScriptable - getValue(String name)**

Returns the named value of the JSON request, instance, or portal.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the JSON request, instance, or portal.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Value of the specified parameter. Null if the specified entity does not exist or has no such parameter.</td>
</tr>
</tbody>
</table>

```javascript
//Server script
(function() {
    data.title = $sp.getValue("title");
    data.catalog = $sp.getValue("sc_catalog");
})();

//HTML template
<div>
    <h1>sp_instance.title: {{::data.title}}</h1>
    <h1>sp_portal.sc_catalog: {{::data.catalog}}</h1>
</div>
```

**Scoped GlideSPScriptable - getValues(Object data, String names)**

Copies values from the request or instance to the data parameter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Object</td>
<td>Receives the parameter values.</td>
</tr>
</tbody>
</table>
ServiceNow  Kingston  Now Platform Custom Business Applications

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>names</td>
<td>String</td>
<td>Comma-separated string of field names.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Scoped GlideSPScriptable - getVariablesArray(Boolean includeNilResponses)**

Returns an array of Service Catalog variables associated with the record in the URL.

For example, if the URL includes the parameters

```
id=form&table=sc_req_item&sys_id=832e9620db4f330083766b984b9619cf
```

the API returns the variables associated with the given record in the Requested item (sc_req_item) table. Must be a record with Service Catalog variables defined, such as a requested item (sc_req_item) record or an incident submitted through a record producer.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeNilResponses</td>
<td>Boolean</td>
<td>Optional. If true, the API includes variables with no user-defined value.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Array of variables associated with the table.</td>
</tr>
</tbody>
</table>

```javascript
//Server script
(function() {
  var rec = $sp.getRecord();
  data.scVars = $sp.getVariablesArray();
})();
```

```html
//HTML template
<div>
  Current record variables: {{::data.scVars}}
</div>
```

**Output:**

```
Current record variables: [
  {
    "display_value":"unlimited",
    "name":"data","visible_summary":true,"id":"472e5620db4f330083766b984b96198a",
    "label":"Data",
    "type":6,
  }
]```
### Scoped GlideSPScriptable - `getWidget(String sysID, Object options)`

Gets a widget by id or sys_id, executes that widget's server script using the provided options, then returns the widget model.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysID</td>
<td>String</td>
<td>The widget sys_id or widget_id.</td>
</tr>
<tr>
<td>options</td>
<td>Object</td>
<td>An object to pass to the widget's server script. Refer to this object as options in your server script.</td>
</tr>
</tbody>
</table>

**Note:** Any options passed into this function will only be available in the embedded widget's server script on the first execution of that script. Any subsequent calls into the server script from the embedded widget will not contain the object properties passed in.

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>A widget model to be used with sp-widget.</td>
</tr>
</tbody>
</table>

```java
//Server script
data.myWidget = $sp.getWidget('widget_id', {p1: param1, p2: param2});
```
Scoped GlideSPScriptable - mapUrlToSPUrl(String url)

Transforms a URL requesting a list or form in the platform UI into the URL of the corresponding id=list or id=form Service Portal page.

Use this method to perform tasks such as redirecting a user after login to the correct Service Portal page when they request a platform UI form or list URL. Note that the id=list and id=form page targets are not customizable.

Note: Table, sys_id, and sysparm_query values are preserved from the original URL; sysparm_view is not.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>String</td>
<td>Platform UI URL</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Transformed Service Portal URL. If the passed-in URL does not request a list or a form in the platform UI, a null value is returned.</td>
</tr>
</tbody>
</table>

List example

```java
GlideSPScriptable().mapUrlToSPUrl("http://demo.service-now.com/task_list.do?
sysparm_userpref_module=1523b8d4c611227b00be8216ec331b9a&sysparm_query=assigned_to=javascript:getMyAssignments()&sysparm_clear_stack=true")
```

Returns

```java
id=list&table=task&filter=assigned_to=javascript:getMyAssignments()&sys_id=&v= 
```

Form example

```java
GlideSPScriptable().mapUrlToSPUrl("incident.do?
sys_id=12bc12bc12bc12bc12bc12bc12bc")
```

Returns

```java
id=form&table=incident&filter=&sys_id=12bc12bc12bc12bc12bc12bc12bc12bc&v= 
```
GlideSysAttachment

The GlideSysAttachment API provides a way to handle attachments.

Content is returned as a string, not as a byte array when `getContents()` is called.
Content is returned as a GlideScriptableInputStream object when `getContentsStream()` is called.
The GlideScriptableInputStream contains the actual bytes not converted into a String.

Scoped GlideSysAttachment - `copy(String sourceTable, String sourceID, String targetTable, String targetID)`

Copies attachments from the source record to the target record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceTable</td>
<td>String</td>
<td>Name of the table with the attachments to be copied.</td>
</tr>
<tr>
<td>sourceID</td>
<td>String</td>
<td>The source table's sysID.</td>
</tr>
<tr>
<td>targetTable</td>
<td>String</td>
<td>Name of the table to have the attachments added.</td>
</tr>
<tr>
<td>targetID</td>
<td>String</td>
<td>The target table's sysID.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Array of sysIDs of the attachments that were copied.</td>
</tr>
</tbody>
</table>

Scoped GlideSysAttachment - `deleteAttachment(String attachmentID)`

Deletes the specified attachment.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attachmentID</td>
<td>String</td>
<td>The attachment's sysID.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Scoped GlideSysAttachment - `getAttachments(String tableName, String sys_id)`

Returns a GlideRecord containing the matching attachment metadata such as name, type, or size.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Name of the table to which the attachment belongs; for example, incident.</td>
</tr>
<tr>
<td>sys_id</td>
<td>String</td>
<td>The sys_id of record to which the attachment belongs.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>GlideRecord object containing the matching attachment metadata such as name, type, or size.</td>
</tr>
</tbody>
</table>

The following script lists attachment file names for a record with two attachments.

```java
var attachment = new GlideSysAttachment();
var attachment = new GlideSysAttachment();
var agr = attachment.getAttachments('<table_name>', '<record_sys_id>');
while(agr.next())
gs.info(agr.getValue('file_name'));
```

Output:

```txt
*** Script: filename1.txt
*** Script: filename2.txt
```

Scoped GlideSysAttachment - getContent(GlideRecord sysAttachment)

Returns the attachment content as a string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysAttachment</td>
<td>GlideRecord</td>
<td>The attachment record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The attachment contents as a string. Returns up to 5 MB of data.</td>
</tr>
</tbody>
</table>
Scoped GlideSysAttachment - getContentBase64( GlideRecord sysAttachment)
Returns the attachment content as a string with base64 encoding.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysAttachment</td>
<td>GlideRecord</td>
<td>The attachment record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The attachment contents as a string with base64 encoding. Returns up to 5 MB of data.</td>
</tr>
</tbody>
</table>

Scoped GlideSysAttachment - getContentStream( String sysID)
Returns a GlideScriptableInputStream object given the sysID of an attachment.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysID</td>
<td>String</td>
<td>The attachment sysID.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideScriptableInputStream</td>
<td>A stream that contains the attachment content.</td>
</tr>
</tbody>
</table>

Scoped GlideSysAttachment - GlideSysAttachment()
Creates an instance of the GlideSysAttachment class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scoped GlideSysAttachment - write(GlideRecord record, String fileName, String contentType, String content)
Inserts an attachment for the specified record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>GlideRecord</td>
<td>The record to which the attachment is to be attached.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>String</td>
<td>The attachment's file name.</td>
</tr>
<tr>
<td>contentType</td>
<td>String</td>
<td>The attachment's content type.</td>
</tr>
<tr>
<td>content</td>
<td>String</td>
<td>The attachment content.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The attachment's sysID. Returns null if the attachment was not added.</td>
</tr>
</tbody>
</table>

**Scoped GlideSysAttachment - writeBase64(GlideRecord gr, String fileName, String contentType, String content_base64Encoded)**

Inserts an attachment for the specified record using base64 encoded content.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>The record to which the attachment is to be attached.</td>
</tr>
<tr>
<td>fileName</td>
<td>String</td>
<td>The attachment's file name.</td>
</tr>
<tr>
<td>contentType</td>
<td>String</td>
<td>The attachment's content type.</td>
</tr>
<tr>
<td>content</td>
<td>String</td>
<td>The attachment content in base64 format.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sysID of the attachment created.</td>
</tr>
</tbody>
</table>

**Scoped GlideSysAttachment - writeContentStream(GlideRecord gr, String fileName, String contentType, GlideScriptableInputStream inputStream)**

Inserts an attachment using the input stream.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>The record to which the attachment is to be attached.</td>
</tr>
<tr>
<td>fileName</td>
<td>String</td>
<td>The attachment's file name.</td>
</tr>
<tr>
<td>contentType</td>
<td>String</td>
<td>The attachment's content type.</td>
</tr>
<tr>
<td>content</td>
<td>GlideScriptableInputStream</td>
<td>The attachment content.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>The sysID of the attachment created.</td>
<td></td>
</tr>
</tbody>
</table>

**GlideSysListControl**

The scoped GlideSysListControl class allows you to determine if the New or Edit buttons are displayed.

**GlideSysListControl - getControlID()**

Returns the sys_id for the control.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>sys_id of the control</td>
</tr>
</tbody>
</table>

```javascript
var sysListCtrl = new GlideSysListControl("incident");
var controlID = sysListCtrl.getControlID();
gs.info(controlID);
```

Output:

91a807a60a0a3c74012113e28b47ca2e

**GlideSysListControl - GlideSysListControl(String tableName)**

Instantiates a GlideSysListControl object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Name of the table</td>
</tr>
</tbody>
</table>

**GlideSysListControl - isOmitEditButton()**

Returns true if the edit button is not displayed.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True when the edit button is not displayed.</td>
</tr>
</tbody>
</table>

```javascript
var sysListCtrl = new GlideSysListControl("incident");
var isOmitted = sysListCtrl.isOmitEditButton();
gs.info(isOmitted);
```

Output:

false

GlideSysListControl - isOmitNewButton()

Returns true when the New button is not displayed.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True when the new button is not displayed.</td>
</tr>
</tbody>
</table>

```javascript
var sysListCtrl = new GlideSysListControl("incident");
var isOmitted = sysListCtrl.isOmitNewButton();
gs.info(isOmitted);
```

Output:

false
GlideSystem

The GlideSystem (referred to by the variable name ‘gs’ in any server-side JavaScript) API provides a number of convenient methods to get information about the system, the current logged in user, etc.

Many of the GlideSystem methods facilitate the easy inclusion of dates in query ranges, and are most often used in filters and reporting.

GlideSystem - addErrorMessage(Object message)

Adds an error message for the current session.

Use `getErrorMessages()` to retrieve a list of error messages currently being shown.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>Object</td>
<td>The message to add.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
gs.include("PrototypeServer");
var ValidatePasswordStronger = Class.create();
ValidatePasswordStronger.prototype = {
    process : function() {
        var user_password = request.getParameter("user_password");
        var min_len = 8;
        var rules = "Password must be at least " + min_len + " characters long and contain a digit, an uppercase letter, and a lowercase letter."
        if (user_password.length() < min_len) {
            gs.addErrorMessage("TOO SHORT: " + rules);
            return false;
        }
        var digit_pattern = new RegExp("[0-9]", "g");
        if (!digit_pattern.test(user_password)) {
            gs.addErrorMessage("DIGIT MISSING: " + rules);
            return false;
        }
        var upper_pattern = new RegExp("[A-Z]", "g");
        if (!upper_pattern.test(user_password)) {
            gs.addErrorMessage("UPPERCASE MISSING: " + rules);
            return false;
        }
        var lower_pattern = new RegExp("[a-z]", "g");
        if (!lower_pattern.test(user_password)) {
            gs.addErrorMessage("LOWERCASE MISSING: " + rules);
            return false;
        }
        return true; // password is OK
    }
};
```
Scoped equivalent

To use the `addErrorMessage()` method in a scoped application, use the corresponding scoped method: `addErrorMessage()`.

**GlideSystem - addInfoMessage(Object message)**

Adds an info message for the current session.

Use `getInfoMessages()` to retrieve the list of info messages being shown. This method is not supported for asynchronous business rules and cannot be used within transform scripts.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>Object</td>
<td>The message to add.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
if (!((current.u_date1.nil()) && (!current.u_date2.nil()))) {
    var start = current.u_date1.getGlideObject().getNumericValue();
    var end = current.u_date2.getGlideObject().getNumericValue();
    if (start > end) {
        gs.addInfoMessage('start must be before end');
        current.u_date1.setError('start must be before end');
        current.setAbortAction(true);
    }
}
```

Scoped equivalent

To use the `addInfoMessage()` method in a scoped application, use the corresponding scoped method: `addInfoMessage()`.

**GlideSystem - addMessage(String type, Object message)**

Adds a message for the current session.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Type of message</td>
</tr>
</tbody>
</table>
### GlideSystem - `beginningOfLastMonth()`

Returns the date and time for the beginning of last month in GMT.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT beginning of last month, in the format <code>yyyy-mm-dd hh:mm:ss</code>.</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `beginningOfLastMonth()` method in a scoped application, use the corresponding scoped method: `beginningOfLastMonth()`.

### GlideSystem - `beginningOfLastWeek()`

Returns the date and time for the beginning of last week in GMT.
### GlideSystem - beginningOfLastWeek()

Returns the date and time for the beginning of last week in GMT.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT beginning of last week, in the format <code>yyyy-mm-dd hh:mm:ss</code>.</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `beginningOfLastWeek()` method in a scoped application, use the corresponding scoped method: `beginningOfLastWeek()`.

### GlideSystem - beginningOfNextWeek()

Returns the date and time for the beginning of next week in GMT.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT beginning of next week, in the format <code>yyyy-mm-dd hh:mm:ss</code>.</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `beginningOfNextWeek()` method in a scoped application, use the corresponding scoped method: `beginningOfNextWeek()`.

### GlideSystem - beginningOfNextMonth()

Returns the date and time for the beginning of next month in GMT.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT beginning of next month, in the format <code>yyyy-mm-dd hh:mm:ss</code>.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT beginning of next month, in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `beginningOfNextMonth()` method in a scoped application, use the corresponding scoped method: `beginningOfNextMonth()`.

GlideSystem - `beginningOfNextYear()`

Returns the date and time for the beginning of next year in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT beginning of next year, in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `beginningOfNextYear()` method in a scoped application, use the corresponding scoped method: `beginningOfNextYear()`.

GlideSystem - `beginningOfThisMonth()`

Returns the date and time for the beginning of this month in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT beginning of this month, in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>
Scoped equivalent

To use the `beginningOfThisMonth()` method in a scoped application, use the corresponding scoped method: `beginningOfThisMonth()`.

**GlideSystem - beginningOfThisQuarter()**

Returns the date and time for the beginning of this quarter in GMT.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>String</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `beginningOfThisQuarter()` method in a scoped application, use the corresponding scoped method: `beginningOfThisQuarter()`.

**GlideSystem - beginningOfThisWeek()**

Returns the date and time for the beginning of this week in GMT.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>String</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `beginningOfThisWeek()` method in a scoped application, use the corresponding scoped method: `beginningOfThisWeek()`.

**GlideSystem - beginningOfThisYear()**

Returns the date and time for the beginning of this year in GMT.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT beginning of this year, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `beginningOfThisYear()` method in a scoped application, use the corresponding scoped method: `beginningOfThisYear()`.

GlideSystem - `beginningOfToday()`

Retrieves the date and time for the beginning of today in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT beginning of today, in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

```javascript
var beginToday = gs.beginningOfToday();
var endToday = gs.endOfToday();
var beginTomorrow = gs.beginningOfTomorrow();
var endTomorrow = gs.endOfTomorrow();
var beginYesterday = gs.beginningOfYesterday();
var endYesterday = gs.endOfYesterday();

gs.info('beginningOfToday: ' + beginToday);
gs.info('endOfToday: ' + endToday);
gs.info('-----------');
gs.info('beginningOfTomorrow: ' + beginTomorrow);
gs.info('endOfTomorrow: ' + endTomorrow);
gs.info('beginningOfYesterday: ' + beginYesterday);
gs.info('endOfYesterday: ' + endYesterday);
gs.info('-----------');
var gdt1 = new GlideDateTime(beginToday);
var gdt2 = new GlideDateTime(beginTomorrow);
var dur = GlideDateTime.subtract(gdt1, gdt2); //the difference between gdt1 and gdt2
```
var gdt1 = new GlideDateTime(endToday);
var gdt2 = new GlideDateTime(endTomorrow);
var dur = GlideDateTime.subtract(gdt1, gdt2); //the difference between gdt1 and gdt2
gs.info('EndTomorrow: ' + dur.getDisplayValue());

var gdt1 = new GlideDateTime(beginToday);
var gdt2 = new GlideDateTime(beginYesterday);
var dur = GlideDateTime.subtract(gdt2, gdt1); //the difference between gdt1 and gdt2
gs.info('BeginningYesterday: ' + dur.getDisplayValue());

var gdt1 = new GlideDateTime(endToday);
var gdt2 = new GlideDateTime(endYesterday);
var dur = GlideDateTime.subtract(gdt2, gdt1); //the difference between gdt1 and gdt2
gs.info('EndYesterday: ' + dur.getDisplayValue());

GlideSystem - beginningOfTomorrow()
Retrieves the (UTC) beginning of tomorrow adjusted for the timezone of the current session.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

var today = new GlideDateTime(gs.beginningOfTomorrow()).getNumericValue();

GlideSystem - beginningOfYesterday()
Returns the date and time for the beginning of yesterday in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT beginning of yesterday, in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

```javascript
var start = new GlideDateTime();
start.setValue(gs.beginningOfYesterday());
```

**GlideSystem - calDateDiff(String startDate, String endDate, Boolean numericValue)**

Calculate the difference between two dates using the default calendar.

Calendars are now legacy. If Schedules are being used, see the topic Calculate Duration Given a Schedule.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startDate</td>
<td>String</td>
<td>The starting date to compare in the current user’s date format.</td>
</tr>
<tr>
<td>endDate</td>
<td>String</td>
<td>The ending date to compare in the current user’s date format.</td>
</tr>
<tr>
<td>numericValue</td>
<td>Boolean</td>
<td>If true, the return value will be formatted in number of seconds; if false, the return value will be formatted ddd hh:mm:ss.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>If the numericValue parameter is true, returns the difference between the two dates as an integer number of seconds; if false, returns the difference between the two dates in the format ddd hh:mm:ss.</td>
</tr>
</tbody>
</table>

**GlideSystem - dateDiff(String startDate, String endDate, Boolean numericValue)**

Calculates the difference between two dates.

This method expects the earlier date as the first parameter and the later date as the second parameter; otherwise, the method returns the difference as a negative value. Use `getDisplayValue()` to convert the strings to the expected format.

This method expects parameters in the user/system date time format, which may not be the same as the internal format. Using parameters in formats other than the user/system date time format may return invalid results.

If you are working with GlideDateTime objects use the GlideDateTime `subtract()` method instead of `dateDiff()`.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startDate</td>
<td>String</td>
<td>The starting date to compare in the current user’s date format.</td>
</tr>
<tr>
<td>endDate</td>
<td>String</td>
<td>The ending date to compare in the current user’s date format.</td>
</tr>
<tr>
<td>numericValue</td>
<td>Boolean</td>
<td>If true, the return value will be formatted in number of seconds; if false the return value will be formatted ddd hh:mm:ss.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>If the numericValue parameter is true, returns the difference between the two dates as an integer number of seconds; if false, returns the difference between the two dates in the format ddd hh:mm:ss.</td>
</tr>
</tbody>
</table>

```javascript
// Given two date/times as DateTime objects
// Set the values this way to ensure a consistent input time
var date1 = new GlideDateTime();
var date2 = new GlideDateTime();
date1.setDisplayValueInternal('2014-01-01 12:00:00');
date2.setDisplayValueInternal('2014-01-01 13:00:00');

// Determine the difference as number of seconds (returns a string)
// Use getDisplayValue() to convert the string to the format expected by dateDiff()
var diffSeconds = gs.dateDiff(date1.getDisplayValue(), date2.getDisplayValue(), true);

// JavaScript will coerce diffSeconds from a string to a number
// since diffSeconds is being compared to a number
var msg = (diffSeconds <= 0) ? ' is on or after ' : ' is before ';
gs.print(date1.getDisplayValue() + msg + date2.getDisplayValue())
```

### GlideSystem - dateGenerate(String date, String range)

Generates a date and time for the specified date in GMT.

To use the `dateGenerate()` method in a scoped application, use the corresponding scoped method: `dateGenerate()`. 
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>String</td>
<td>Format: yyyy-mm-dd.</td>
</tr>
<tr>
<td>range</td>
<td>String</td>
<td>Start, end, or a time in the 24-hour format hh:mm:ss.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Date and time in the format yyyy-mm-dd hh:mm:ss. If range is start, the return value is yyyy-mm-dd 00:00:00. If range is end, the return value is yyyy-mm-dd 23:59:59.</td>
</tr>
</tbody>
</table>

GlideSystem - daysAgo(Number days)

Returns a date and time for a certain number of days ago.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>Number</td>
<td>Number of days</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

```javascript
function contractNoticeDue() {
  var gr = new GlideRecord("contract");
  gr.addQuery("u_contract_status", "Active");
  gr.query();
  while (gr.next()) {
    if ((gr.u_termination_date <= gs.daysAgo(-90)) &&
    (gr.u_contract_duration == "Long")) {
      gr.u_contract_status = "In review";
    } else if ((gr.u_termination_date <= gs.daysAgo(-50)) &&
    (gr.u_contract_duration == "Medium")) {
      gr.u_contract_status = "In review";
    } else if ((gr.u_termination_date <= gs.daysAgo(-10)) &&
    (gr.u_contract_duration == "Short")) {
      gr.u_contract_status = "In review";
    }
    gr.update();
  }
}
Scoped equivalent

To use the `daysAgo()` method in a scoped application, use the corresponding scoped method: `daysAgo0`.

**GlideSystem - `daysAgoEnd(Number days)`**

Returns a date and time for the end of the day a specified number of days ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>Number</td>
<td>Integer number of days</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT end of the day in the format <code>yyyy-mm-dd hh:mm:ss</code>.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `daysAgoEnd()` method in a scoped application, use the corresponding scoped method: `daysAgoEnd0`.

**GlideSystem - `daysAgoLocal(Number days)`**

Returns the date and time of the beginning of the day for the specified number of days ago. The returned date and time reflect the time zone of the current session (local time).

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>Number</td>
<td>Number of days ago</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Local date and time for the beginning of the day in the user-defined date time format. If the date time format is not modified from its initial value the format is <code>yyyy-mm-dd hh:mm:ss</code>.</td>
</tr>
</tbody>
</table>

```java
if (due_in == "1 Day") {
    dd = gs.daysAgoLocal(-1);
} else if (due_in == "1 Week") {
    dd = gs.daysAgoLocal(-7);
```
GlideSystem - daysAgoStart(Number days)

Returns a date and time for the beginning of the day a specified number of days ago.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>String</td>
<td>Integer number of days</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT start of the day in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

var gr = new GlideRecord('sysapproval_approver');
gr.addQuery('state', 'requested');
gr.addQuery('sys_updated_on', '<', gs.daysAgoStart(5));
gr.query();

Scoped equivalent

To use the daysAgoStart() method in a scoped application, use the corresponding scoped method: daysAgoStart().

GlideSystem - endOfLastMonth()

Returns the date and time for the end of last month in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT end of last month, in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>
**Scoped equivalent**

To use the `endOfLastMonth()` method in a scoped application, use the corresponding scoped method: `endOfLastMonth()`.

**GlideSystem - endOfLastWeek()**

Returns the date and time for the end of last week in GMT.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT end of last week, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `endOfLastWeek()` method in a scoped application, use the corresponding scoped method: `endOfLastWeek()`.

**GlideSystem - endOfLastYear()**

Returns the date and time for the end of last year in GMT.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `endOfLastYear()` method in a scoped application, use the corresponding scoped method: `endOfLastYear()`.

**GlideSystem - endOfNextMonth()**

Returns the date and time for the end of next month in GMT.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `endOfNextMonth()` method in a scoped application, use the corresponding scoped method: `endOfNextMonth()`.

**GlideSystem - endOfNextWeek()**

Returns the date and time for the end of next week in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `endOfNextWeek()` method in a scoped application, use the corresponding scoped method: `endOfNextWeek()`.

**GlideSystem - endOfNextYear()**

Returns the date and time for the end of next year in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `endOfNextYear()` method in a scoped application, use the corresponding scoped method: `endOfNextYear()`.

GlideSystem - `endOfThisMonth()`

Returns the date and time for the end of this month in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `endOfThisMonth()` method in a scoped application, use the corresponding scoped method: `endOfThisMonth()`.

GlideSystem - `endOfThisQuarter()`

Returns the date and time for the end of this quarter in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>
Scoped equivalent

To use the `endOfThisQuarter()` method in a scoped application, use the corresponding scoped method: `endOfThisQuarter()`.

**GlideSystem - endOfThisWeek()**

Returns the date and time for the end of this week in GMT.

<table>
<thead>
<tr>
<th>Parameters</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
<td></td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `endOfThisWeek()` method in a scoped application, use the corresponding scoped method: `endOfThisWeek()`.

**GlideSystem - endOfThisYear()**

Returns the date and time for the end of this year in GMT.

<table>
<thead>
<tr>
<th>Parameters</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss.</td>
<td></td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `endOfThisYear()` method in a scoped application, use the corresponding scoped method: `endOfThisYear()`.

**GlideSystem - endOfDay()**

Returns the date and time for the end of today in GMT.
GlideSystem - endOfTomorrow()

Retrieves the date and time for the end of tomorrow in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

var tomorrowEnd = new GlideDateTime();
tomorrowEnd.setValue(gs.endOfTomorrow());

GlideSystem - endOfYesterday()

Gets the date and time for the end of yesterday in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

GlideSystem - eventQueue(String name, Object glideRecord, String parm1, String parm2, String queue)

Queues an event for the event manager.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the event being queued.</td>
</tr>
<tr>
<td>glideRecord</td>
<td>Object</td>
<td>GlideRecord object, such as &quot;current&quot;.</td>
</tr>
<tr>
<td>parm1</td>
<td>String</td>
<td>(Optional) Saved with the instance if specified.</td>
</tr>
<tr>
<td>parm2</td>
<td>String</td>
<td>(Optional) Saved with the instance if specified.</td>
</tr>
<tr>
<td>queue</td>
<td>String</td>
<td>Name of the queue.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
if (current.operation() != 'insert' &&
current.comments.changes()) {
    gs.eventQueue("incident.commented", current, gs.getUserID(),
    gs.getUserName());
}
```

**Scoped equivalent**

To use the `eventQueue()` method in a scoped application, use the corresponding scoped method: `eventQueue()`.

### GlideSystem - `eventQueueScheduled(String name, Object glideRecord, String parm1, String parm2, Object expiration)`

Queues an event for the event manager at a specified date and time.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the event being queued.</td>
</tr>
<tr>
<td>glideRecord</td>
<td>Object</td>
<td>GlideRecord object, such as &quot;current&quot;.</td>
</tr>
<tr>
<td>parm1</td>
<td>String</td>
<td>(Optional) Saved with the instance if specified.</td>
</tr>
<tr>
<td>parm2</td>
<td>String</td>
<td>(Optional) Saved with the instance if specified.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.

ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
### eventQueueScheduled()

If (current.operation() != 'insert' &&
    current.comments.changes()) {
    gs.eventQueueScheduled("incident.commented", current,
        gs.getUserID(), gs.getUserName(), new GlideDateTime('2018-06-02
        20:00:00'));
}

### Scoped equivalent

To use the `eventQueueScheduled()` method in a scoped application, use the corresponding scoped method: `eventQueueScheduled()`.

### GlideSystem - flushMessages()

Clears session messages saved using `addErrorMessage()` or `addInfoMessage()`.

Session messages are shown at the top of the form. In client side scripts, use `g_form.clearMessages()` to remove session messages.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
gs.flushMessages();
```

### GlideSystem - getAvatar()

Returns the file path to the user's avatar.
### GlideSystem - `getUser().getAvatar()`

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The file path to the user's avatar.</td>
</tr>
</tbody>
</table>

```javascript
var avatarFile = gs.getUser().getAvatar();
gs.addInfoMessage('User avatar ID: ' + avatarFile);
```

Output:

```
User avatar ID: c148e1d13741310042106710ce41f149.iix?t=small
```

### GlideSystem - `getCurrentScopeName()`

Returns the name of the current scope.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current scope name</td>
</tr>
</tbody>
</table>

```javascript
gs.getCurrentScopeName();
```

### Scoped equivalent

To use the `getCurrentScopeName()` method in a scoped application, use the corresponding scoped method: `getCurrentScopeName()`.

### GlideSystem - `getDisplayColumn(String tableName)`

Returns the display column for the table.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Name of the table from which to get the display column name.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Display column name</td>
</tr>
</tbody>
</table>

```javascript
// Return the sys_id value for a given table and its display value
function GetIDValue(table, displayValue) {
  var rec = new GlideRecord(table);
  var dn = gs.getDisplayColumn(table);
  if (rec.get(dn, displayValue))
    return rec.sys_id;
  else
    return null;
}
```

**GlideSystem - getDisplayValueFor(String tableName, String recordID, String fieldName)**

Returns the display value for a specified field on a record.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Name of the table</td>
</tr>
<tr>
<td>recordID</td>
<td>String</td>
<td>sysid for the record</td>
</tr>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Display value for the field</td>
</tr>
</tbody>
</table>

### GlideSystem - getErrorMessages()

Returns the list of error messages for the session that were added by `addErrorMessage()`.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>List of error messages</td>
</tr>
</tbody>
</table>

### Scoped equivalent

To use the `getErrorMessages()` method in a scoped application, use the corresponding scoped method: `getErrorMessages()`.

### GlideSystem - `getEscapedProperty(String key, Object substituteObject)`

Gets the property and escapes it for XML parsing.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>String</td>
<td>Key for the property whose value should be returned.</td>
</tr>
<tr>
<td>substituteObject</td>
<td>Object</td>
<td>Object to be returned if the property is not found.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The property, or the substituteObject if the property is not found.</td>
</tr>
</tbody>
</table>

### GlideSystem - `getImpersonatingUserDisplayName()`

Returns the display name of the impersonating user.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The display name of the impersonating user</td>
</tr>
</tbody>
</table>

### GlideSystem - `getImpersonatingUserName()`

Returns the name of the impersonating user or null if not impersonating.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Name of impersonating user</td>
</tr>
</tbody>
</table>

**GlideSystem - getInfoMessages()**

Returns the list of info messages for the session that were added by `addInfoMessage()`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>List of info messages</td>
</tr>
</tbody>
</table>

**GlideSystem - getInitials()**

Returns the user's initials.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The user's initials.</td>
</tr>
</tbody>
</table>

```javascript
var userInitials = gs.getUser().getInitials();
gs.addInfoMessage('User initials: ' + userInitials);
```

Output:

```
User initials: SA
```
GlideSystem - getMessage(String messageID, Object args)

Retrieves translated messages to display in the UI.

If the specified string exists in the database for the current language, then the translated message is returned. If the specified string does not exist for the current language, then the English version of the string is returned. If the string does not exist at all in the database, then the ID itself is returned.

If the UI message has a tick (©), there may be issues with the message in the script; to escape the ticks (©), use getMessageS(String, Object).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageID</td>
<td>String</td>
<td>The ID of the message.</td>
</tr>
<tr>
<td>args</td>
<td>Object</td>
<td>(Optional) a list of strings or other values defined by java.text.MessageFormat, which allows you to produce language-neutral messages for display to users.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The UI message</td>
</tr>
</tbody>
</table>

var my_message = '${gs.getMessage("This is a message.")}';
alert(my_message);

Using the optional parameter:

```
gs.getMessage("\"{0}\" is not a Client Callable Script Include\", 'BAR');
```

Output:

"BAR" is not a Client Callable Script Include

Scoped equivalent

To use the getMessage() method in a scoped application, use the corresponding scoped method: getMessage().
GlideSystem - getMessages(String type)

Returns the list of messages of the specified type for the session that were added by addMessage().

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>The type of message</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideSystem - getMessageS(String messageID, Object args)

Retrieves translated messages to display in the UI and escapes all ticks ('). If the specified string exists in the database for the current language, then the translated message is returned. If the specified string does not exist for the current language, then the English version of the string is returned. If the string does not exist at all in the database, then the ID itself is returned. Useful if you are inserting into a JavaScript expression from Jelly.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageID</td>
<td>String</td>
<td>The ID of the message.</td>
</tr>
<tr>
<td>args</td>
<td>Object</td>
<td>(Optional) a list of strings or other values defined by java.text.MessageFormat, which allows you to produce language-neutral messages for display to users.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message with ticks escaped.</td>
</tr>
</tbody>
</table>

```
/*
 "I love France" translates to "J'aime la France" in French. Rendering this without escaping the tick in "J'aime" would break Jelly, because the tick would prematurely end the variable assignment, and everything that follows (aime la France') would be a jelly syntax error.
 */
var my_message = '${gs.getMessageS("I love France")}';
```
alert(my_message);

Output:

J’aime la France

**GlideSystem - getNodeValue(Object obj, Number index)**

Returns the node value for specified index.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>Object</td>
<td>Object to examine</td>
</tr>
<tr>
<td>index</td>
<td>Number</td>
<td>Index from which to get the node value</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The node’s value</td>
</tr>
</tbody>
</table>

**GlideSystem - getNodeName(Object obj, Number index)**

Returns the node name for specified index.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>Object</td>
<td>Object to examine</td>
</tr>
<tr>
<td>index</td>
<td>Number</td>
<td>Index from which to get the node name</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The node’s name</td>
</tr>
</tbody>
</table>

**GlideSystem - getPreference(String key, Object default)**

Returns a user preference.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>String</td>
<td>The key for the preference</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>default</td>
<td>Object</td>
<td>The default value</td>
</tr>
</tbody>
</table>

### GlideSystem - getProperty(String key, Object alt)

Returns the value of a Glide property. If the property is not found, returns the alternate value.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>String</td>
<td>The key for the property whose value should be returned.</td>
</tr>
<tr>
<td>alt</td>
<td>Object</td>
<td>(Optional) Alternate object to return if the property is not found.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the Glide property, or the alternate object defined above.</td>
</tr>
</tbody>
</table>

```javascript
//Check for attachments and add link if there are any
var attachment_link = '';
var rec = new GlideRecord('sc_req_item');
rec.addQuery('sys_id', current.request_item);
rec.query();
if(rec.next()){
  if(rec.hasAttachments()){
    attachment_link = gs.getProperty('glide.servlet.uri') + rec.getLink();
  }
}
```

### Scoped equivalent

To use the `getProperty()` method in a scoped application, use the corresponding scoped method: `getProperty0`.

### GlideSystem - getScriptError(String script)

Returns the script error found in the specified script, if there is one.
The script is not executed by this function, only checked for syntax errors.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>script</td>
<td>String</td>
<td>The script to check for errors.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The error message, or null if there is no error.</td>
</tr>
</tbody>
</table>

**GlideSystem - getSession()**

Returns a GlideSession object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideSession Object</td>
<td>A GlideSession object for the current session</td>
</tr>
</tbody>
</table>

```javascript
var session = gs.getSession();
```

**Scoped equivalent**

To use the `getSession()` method in a scoped application, use the corresponding scoped method: `getSession()`.

**GlideSystem - getSessionID()**

Returns the GlideSession Session ID.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The session ID</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `getSessionID()` method in a scoped application, use the corresponding scoped method: `getSessionID`.

GlideSystem - ```getStyle(String tableName, StringfieldName, String fieldValue)``` 

Returns the style defined for the table, field and value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table name</td>
</tr>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The field name</td>
</tr>
<tr>
<td>fieldValue</td>
<td>String</td>
<td>The field value</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The style</td>
</tr>
</tbody>
</table>

GlideSystem - ```getTrivialMessages()``` 

Returns the list of error messages for the session that were added with the trivial flag.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>List of messages</td>
</tr>
</tbody>
</table>

GlideSystem - ```getUser()``` 

Returns a reference to the user object for the current user.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideUser</td>
<td>A reference to a user object for the current user</td>
</tr>
</tbody>
</table>

```javascript
var myUserObject = gs.getUser()
```

Scoped equivalent
To use the `getUser()` method in a scoped application, use the corresponding scoped method: `getUser()`.

**GlideSystem - getUserDisplayName()**

Returns the name field of the current user (e.g. John Smith, as opposed to smith).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The user's display name</td>
</tr>
</tbody>
</table>

```javascript
gs.getUserDisplayName();
```

Scoped equivalent
To use the `getUserDisplayName()` method in a scoped application, use the corresponding scoped method: `getUserDisplayName()`.

**GlideSystem - getUserID()**

Returns the sys_id of the current user.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the current user.</td>
</tr>
</tbody>
</table>

```java
if (current.operation() != 'insert' &&
    current.comments.changes()) {
    gs.eventQueue("incident.commented", current, gs.getUserID(),
    gs.getUserName());
}
```

Scoped equivalent

To use the `getUserID()` method in a scoped application, use the corresponding scoped method: `getUserID()`.

GlideSystem - `getUserName()`

Returns the username of the current user (e.g., jsmith).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Username of the current user</td>
</tr>
</tbody>
</table>

```java
//Add a comment when closing
    current.comments = "Closed by " + gs.getUserName() + " at "
    + gs.nowDateTime();
    gs.addInfoMessage("Close comment added");
```
 Scoped equivalent

To use the `getUserName()` method in a scoped application, use the corresponding scoped method: `getUserName()`.  

**GlideSystem - getUsernameByUserID(String sys_id)**

Returns the username based on a user ID.

<table>
<thead>
<tr>
<th>Parameters</th>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>A sys_id for a user</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The username</td>
</tr>
</tbody>
</table>

**GlideSystem - getXMLNodeList(String xml)**

Constructs an array of all the nodes and values in an XML document.

<table>
<thead>
<tr>
<th>Parameters</th>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml</td>
<td>String</td>
<td>The XML document to parse</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array list</td>
<td>List of names and values</td>
</tr>
</tbody>
</table>

**GlideSystem - getXMLText(String xml, String xpathQuery)**

Returns the XML text for the first node in the XML string that matches the XPath query.

<table>
<thead>
<tr>
<th>Parameters</th>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml</td>
<td>String</td>
<td>An XML string</td>
</tr>
<tr>
<td>xpathQuery</td>
<td>String</td>
<td>The XPath query to match</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The XML node</td>
</tr>
</tbody>
</table>
GlideSystem - hasRole(String roleName)

Determines if the current user has at least one of the passed-in roles.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roleName</td>
<td>String</td>
<td>Comma separated list of roles</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user has at least one of the passed-in roles; otherwise false. Also returns true if the user has the administrator role.</td>
</tr>
</tbody>
</table>

```java
if (!gs.hasRole("admin, groups_admin") &&
    gs.getSession().isInteractive()) {
    var qc = current.addQuery("u_hidden", "!=", "true"); // cannot see hidden groups...
    qc.addOrCondition("sys_id", "javascript:getMyGroups()"); // ...unless in the hidden group
}
```

Scoped equivalent

To use the hasRole() method in a scoped application, use the corresponding scoped method: hasRole().

GlideSystem - hasRoleInGroup(Object roleName, Object group)

Determines if the current user has the specified role within a specified group.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roleName</td>
<td>Object</td>
<td>Name of the role</td>
</tr>
<tr>
<td>group</td>
<td>Object</td>
<td>sys_id of the group to check for the specified role</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Boolean    | Returns true when the following conditions are met, otherwise returns false.
|            | - The logged-in user is assigned to the specified role.                     |
|            | - The granted by field on the user role record is set to the specified group.|
|            | - The inherited field on the user role record is false.                     |

```javascript
var group = new GlideRecord('sys_user_group');
group.addQuery('name', 'GROUP_NAME');
group.setLimit(1);
group.query();
if (group.next()) {
  if (gs.hasRoleInGroup('role_name', group)) {
    gs.print('User has role in group');
  } else {
    gs.print('User does NOT have role in group');
  }
}
```

GlideSystem - hoursAgo(Number hours)

Returns a date and time for a certain number of hours ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hours</td>
<td>Number</td>
<td>Number of hours</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

```javascript
if (current.operation() == 'insert') {
  // If no due date was specified, calculate a default
  if (current.due_date == '') {
    if (current.urgency == '1') {
      // Set due date to 4 hours ahead of current time
      current.due_date = gs.hoursAgo(-4);
    }
    if (current.urgency == '2') {
      // Set due date to 8 hours ahead of current time
      current.due_date = gs.hoursAgo(-8);
    }
  }
```
// Set due date to 2 days ahead of current time
    current.due_date = gs.daysAgo(-2);
}

if (current.urgency == '3') {
    // Set due date to 7 days ahead of current time
    current.due_date = gs.daysAgo(-7);
}
}

**Scoped equivalent**

To use the `hoursAgo()` method in a scoped application, use the corresponding scoped method: `hoursAgo()`.

**GlideSystem - hoursAgoEnd(Number hours)**

Returns a date and time for the end of the hour a certain number of hours ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hours</td>
<td>Number</td>
<td>An integer number of hours</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `hoursAgoEnd()` method in a scoped application, use the corresponding scoped method: `hoursAgoEnd()`.

**GlideSystem - hoursAgoStart(Number hours)**

Returns a date and time for the start of the hour a certain number of hours ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hours</td>
<td>Number</td>
<td>An integer number of hours</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>
Scoped equivalent

To use the `hoursAgoStart()` method in a scoped application, use the corresponding scoped method: `hoursAgoStart()`.

**GlideSystem - isFirstDayOfMonth(Object date)**

Checks whether the date is the first day of the month.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Object</td>
<td>date object</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the date is the first day of the month, false otherwise.</td>
</tr>
</tbody>
</table>

**GlideSystem - isFirstDayOfWeek(Object date)**

Checks whether the date is the first day of the week. This uses the ISO standard of Monday being the first day of the week.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Object</td>
<td>date object</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the date is the first day of the month, false otherwise.</td>
</tr>
</tbody>
</table>

**GlideSystem - isFirstDayOfYear(Object date)**

Checks whether the date is the first day of the year.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Object</td>
<td>date object</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.

ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
### GlideSystem - `isInteractive()`

Checks if the current session is interactive.

An example of an interactive session is when a user logs in using the log-in screen. An example of a non-interactive session is using a SOAP request to retrieve data.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the session is interactive.</td>
</tr>
</tbody>
</table>

```java
if (!gs.hasRole('admin') && gs.isInteractive()) { 
    var qcl = current.addQuery('u_group', '');
    var gra = new GlideRecord('sys_user_grmember');
    gra.addQuery('user', gs.getUserID());
    gra.query();
    while (gra.next()) {
        qcl.addOrCondition('u_group', gra.group);
    }
}
```

### Scoped equivalent

To use the `isInteractive()` method in a scoped application, use the corresponding scoped method: `isInteractive0`.

### GlideSystem - `isLastDayOfMonth(Object date)`

Checks whether the date is the last day of the month.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Object</td>
<td>date object</td>
</tr>
</tbody>
</table>
## GlideSystem - isLastDayOfWeek(Object date)

Checks whether the date is the last day of the week.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Object</td>
<td>date object</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the date is the last day of the week, false otherwise.</td>
</tr>
</tbody>
</table>

## GlideSystem - isLastDayOfYear(Object date)

Checks whether the date is the last day of the year.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Object</td>
<td>date object</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if date is the last day of the year, false otherwise.</td>
</tr>
</tbody>
</table>

## GlideSystem - isLoggedIn()

Determines if the current user is currently logged in.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### GlideSystem - isLoggedIn()

**Description**

True if the current user is logged in, false otherwise.

**Parameters**

- None

**Returns**

- **Type**: Boolean
- **Description**: True if the current user is logged in, false otherwise.

**Scoped equivalent**

To use the `isLoggedIn()` method in a scoped application, use the corresponding scoped method: `isLoggedIn()`.

```java
if (gs.isLoggedIn())
    gs.info("submitted from mobile UI");
else
    gs.info("NOT submitted from mobile UI");
```

### GlideSystem - isMobile()

**Description**

Determines whether a request comes from a mobile device. This method can be used in UI action conditions and business rules.

**Parameters**

- None

**Returns**

- **Type**: Boolean
- **Description**: True if the request comes from a mobile device; otherwise, false.

**Scoped equivalent**

To use the `isMobile()` method in a scoped application, use the corresponding scoped method: `isMobile()`.

```java
if (gs.isMobile())
    gs.info("submitted from mobile UI");
else
    gs.info("NOT submitted from mobile UI");
```

### GlideSystem - lastWeek()

Returns the date and time one week ago in GMT.

**Parameters**

- None

**Returns**

- **Type**: DateTime
- **Description**: Returns the date and time one week ago in GMT.

**Scoped equivalent**

To use the `lastWeek()` method in a scoped application, use the corresponding scoped method: `lastWeek()`.
### GlideSystem - log(String message, String source)

Logs a message to the system log and saves it to the syslog table.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to log.</td>
</tr>
<tr>
<td>source</td>
<td>String</td>
<td>(optional) The source of the message.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var count = new GlideAggregate('incident');
count.addQuery('active', 'true');
count.addAggregate('COUNT', 'category');
count.query();
while (count.next()) {
    var category = count.category;
    var categoryCount = count.getAggregate('COUNT', 'category');
    gs.log("The are currently " + categoryCount + " incidents with a category of " + category, "Incident Counter");
}
```

### GlideSystem - logError(String message, String source)

Logs an error to the system log and saves it to the syslog table.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to log.</td>
</tr>
<tr>
<td>source</td>
<td>String</td>
<td>(optional) The source of the message.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideSystem - logWarning(String message, String source)**

Logs a warning to the system log and saves it to the syslog table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to log.</td>
</tr>
<tr>
<td>source</td>
<td>String</td>
<td>(optional) The source of the message</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideSystem - minutesAgo(Number minutes)**

Returns a date and time for a certain number of minutes ago

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>minutes</td>
<td>Number</td>
<td>An integer number of minutes</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

```java
// Check to see if the user has failed to login too many times
// when the limit is reached, lock the user out of the system
//
// Check failed logins in the last 15 minutes
var gr = new GlideRecord('sysevent');
gr.addQuery('name', 'login.failed');
gr.addQuery('parm1', event.parm1.toString());
gr.addQuery('sys_created_on','>=', gs.minutesAgo(15));
gr.query();
var rowCount = gr.getRowCount();
if(rowCount >= 5){
```
var gr = new GlideRecord("sys_user");
gr.addQuery("user_name", event.parm1.toString());
gr.query();
if (gr.next()) {
    gr.locked_out = true;
gr.update();
gs.log("User " + event.parm1 + " locked out due to too many invalid login attempts");
}

**GlideSystem - minutesAgoEnd(NumberOf minutes)**

Returns a date and time for the end of the minute a certain number of minutes ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>minutes</td>
<td>Number</td>
<td>An integer number of minutes</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `minutesAgoEnd()` method in a scoped application, use the corresponding scoped method: `minutesAgoEnd()`.

**GlideSystem - minutesAgoStart(NumberOf minutes)**

Returns a date and time for the start of the minute a certain number of minutes ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>minutes</td>
<td>Number</td>
<td>An integer number of minutes</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `minutesAgoStart()` method in a scoped application, use the corresponding scoped method: `minutesAgoStart()`.
**GlideSystem - monthsAgo(Number months)**

Returns a date and time for a certain number of months ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>months</td>
<td>Number</td>
<td>An integer number of months</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT on today's date of the specified month, in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

**Scoped equivalent**

To use the `monthsAgo()` method in a scoped application, use the corresponding scoped method: `monthsAgo0`.

**GlideSystem - monthsAgoEnd(Number months)**

Returns a date and time for the last day of the month a certain number of months ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>months</td>
<td>Number</td>
<td>An integer number of months</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT end of the month the specified number of months ago, in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

**GlideSystem - monthsAgoStart(Number months)**

Returns a date and time for the start of the month a certain number of months ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>months</td>
<td>Number</td>
<td>An integer number of months</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT start of the month the specified number of months ago, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `monthsAgoStart()` method in a scoped application, use the corresponding scoped method: `monthsAgoStart()`.

GlideSystem - `nil(Object obj)`

Queries an object and returns true if the object is null or contains an empty string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>Object</td>
<td>The object to be checked.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if null or empty string; otherwise, false.</td>
</tr>
</tbody>
</table>

```javascript
if (!current.u_date1.nil() && !current.u_date2.nil()) {
    var start = current.u_date1.getGlideObject().getNumericValue();
    var end = current.u_date2.getGlideObject().getNumericValue();
    if (start > end) {
        gs.addInfoMessage('start must be before end');
        current.u_date1.setError('start must be before end');
        current.setAbortAction(true);
    }
}
```

Scoped equivalent

To use the `nil()` method in a scoped application, use the corresponding scoped method: `nil()`.

GlideSystem - `now()`

Returns the current date in UTC.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current date in the user-defined format, in UTC.</td>
</tr>
</tbody>
</table>

// When the user password changes then set the u_password_last_reset field // to now so we know when to force another update

var gr = new GlideRecord("sys_user");
if (gr.get(event.parm1.toString())) {
    // Do something based on the Password Changing
    gs.log("The user password changed so do something else...");
    gr.u_password_last_reset = gs.now();
    gr.update();
}

GlideSystem - nowNoTZ()

Returns the current date and time in UTC format.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current UTC date time</td>
</tr>
</tbody>
</table>

// When the user password changes then set the u_password_last_reset field // to now so we know when to force another update

var gr = new GlideRecord("sys_user");
if (gr.get(event.parm1.toString())) {
    // Do something based on the Password Changing
    gs.log("The user password changed so do something else...");
    gr.u_password_last_reset = gs.nowNoTZ();
    gr.update();
}
GlideSystem - nowDateTime()

Returns the current date and time in the user-defined format.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current date and time in the user-defined format.</td>
</tr>
</tbody>
</table>

```javascript
current.u_target_date = gs.nowDateTime();
```

When setting a variable in a workflow script to the current date and time, use the `setDisplayValue()` method. The following script sets the workflow variable `end_date` to the current date and time.

```javascript
current.variables.end_date.setDisplayValue(gs.nowDateTime());
```

GlideSystem - nowGlideDateTime()

Returns a GlideDateTime object with the current date and time.

After you get a GlideDateTime object with the current date and time, you can use the GlideDateTime methods to perform date-time operations, such as performing date-time calculations, formatting a date-time, or converting between date-time formats.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>GlideDateTime object with the current date and time in GMT format.</td>
</tr>
</tbody>
</table>
Sets the field u_target_date to the current date and time in GMT format.

```java
current.u_target_date = gs.nowGlideDateTime();
```

**GlideSystem - print(String message)**

Writes a message to the system log.

This method does not write the message to the syslog table unless debug has been activated.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to log</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var rec = new GlideRecord('incident');
rec.addQuery('active',false);
rec.query();
while (rec.next()) {
    gs.print('Inactive incident ' + rec.number + ' deleted');
    rec.deleteRecord();
}
```

**GlideSystem - quartersAgo(Number quarters)**

Gets a date and time for a certain number of quarters ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>quarters</td>
<td>Number</td>
<td>An integer number of quarters</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT beginning of the quarter that was the specified number of quarters ago, in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

**GlideSystem - quartersAgoEnd(Number quarters)**

Returns a date and time for the last day of the quarter, for a specified number of quarters ago.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>quarters</td>
<td>Number</td>
<td>An integer number of quarters</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT end of the quarter that was the specified number of quarters ago, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the quartersAgoEnd() method in a scoped application, use the corresponding scoped method: quartersAgoEnd().

GlideSystem - quartersAgoStart(Number quarters)

Returns a date and time for the first day of the quarter, for a specified number of quarters ago.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>quarters</td>
<td>Number</td>
<td>An integer number of quarters</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT end of the month that was the specified number of quarters ago, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the quartersAgoStart() method in a scoped application, use the corresponding scoped method: quartersAgoStart().

GlideSystem - setProperty(String key, String value, String description)

Sets the specified key to the specified value.

Note: Care should be taken when setting system properties (sys_properties) using this method as it causes a system-wide cache flush. Each flush can cause system degradation while the caches rebuild. If a value must be updated often, it should not be stored as a system property. In general, you should only place values in the sys_properties table that do not frequently change.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>String</td>
<td>The key for the property to be set.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value of the property to be set.</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>A description of the property.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
gs.setProperty("glide.foo","bar","foo");
gs.info(gs.getProperty("glide.foo"));
```

Output: bar

---

**Scoped equivalent**

To use the `setProperty()` method in a scoped application, use the corresponding scoped method: `setProperty()`.  

**GlideSystem - setRedirect(Object URI)**

Sets the redirect URI for this transaction, which then determines the next page the user will see.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>Object</td>
<td>URI to set as the redirect</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

This example redirects the user to a particular catalog item, and passes along the current email as a parameter.

```java
gs.setRedirect("com.glideapp.servicecatalog_cat_item_view.do?sysparm_id=d41ce5bac611227a0167f4bf8109bf70&sysparm_user=" + current.sys_id + "&sysparm_email=" + current.email)
```
Scoped equivalent

To use the `setRedirect()` method in a scoped application, use the corresponding scoped method: `setRedirect()`.

GlideSystem - `setReturn(Object URI)`

Sets the return URI for this transaction. This determines what page the user will be directed to when they return from the next form.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>Object</td>
<td>URI to set as the return location.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

This example ensures that the user will be returned to the current page when they are done with the next one.

```java
gs.setReturn (current.getLink(true));
```

GlideSystem - `tableExists(String tableName)`

Determines if a database table exists.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Name of the table</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the table exists, false otherwise.</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `tableExists()` method in a scoped application, use the corresponding scoped method: `tableExists()`.

GlideSystem - `userID()`

Returns the sys_id of the user associated with this session. Use `get UserID()` instead.
### GlideSystem - workflowFlush(Object glideRecord)

Deletes all existing workflow operations for the specified GlideRecord.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>glideRecord</td>
<td>Object</td>
<td>The GlideRecord</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### GlideSystem - yearsAgo(Number years)

Returns a date and time for a certain number of years ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>years</td>
<td>Number</td>
<td>An integer number of years</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT beginning of the year that is the specified number of years ago, in the format yyyy-mm-dd hh:mm:ss.</td>
</tr>
</tbody>
</table>

---

**Scoped equivalent**

To use the `yearsAgo(Number years)` method in a scoped application, use the corresponding scoped method: `yearsAgo(Number years)`. 

© 2020 ServiceNow, Inc. All rights reserved.

ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
GlideSystem - yesterday()

Returns yesterday's time (24 hours ago).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT for 24 hours ago, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `yesterday()` method in a scoped application, use the corresponding scoped method: `yesterday()`.

GlideSystem

The scoped GlideSystem (referred to by the variable name ‘gs’ in any server-side JavaScript) API provides a number of convenient methods to get information about the system, the current logged in user, etc.

Many of the GlideSystem methods facilitate the easy inclusion of dates in query ranges, and are most often used in filters and reporting.

Scoped GlideSystem - addErrorMessage(Object message)

Adds an error message for the current session.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>Object</td>
<td>The message to add.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
gs.include("PrototypeServer");
var ValidatePasswordStronger = Class.create();
ValidatePasswordStronger.prototype = {
  process : function() {
    var user_password = request.getParameter("user_password");
```
```javascript
var min_len = 8;
var rules = "Password must be at least " + min_len + " characters long and contain a digit, an uppercase letter, and a lowercase letter."
if (user_password.length() < min_len) {
    gs.addErrorMessage("TOO SHORT: " + rules);
    return false;
}
var digit_pattern = new RegExp("[0-9]", "g");
if (!digit_pattern.test(user_password)) {
    gs.addErrorMessage("DIGIT MISSING: " + rules);
    return false;
}
var upper_pattern = new RegExp("[A-Z]", "g");
if (!upper_pattern.test(user_password)) {
    gs.addErrorMessage("UPPERCASE MISSING: " + rules);
    return false;
}
var lower_pattern = new RegExp("[a-z]", "g");
if (!lower_pattern.test(user_password)) {
    gs.addErrorMessage("LOWERCASE MISSING: " + rules);
    return false;
}
return true; // password is OK
}

Scoped GlideSystem - addInfoMessage(Object message)

Adds an info message for the current session. This method is not supported for asynchronous business rules.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>Object</td>
<td>An info message object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
Scoped GlideSystem - base64Encode(String source)

Creates a base64 string from the specified string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>String</td>
<td>The string to be encoded.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The base64 string.</td>
</tr>
</tbody>
</table>

Scoped GlideSystem - base64Decode(String source)

Returns an ASCII string from the specified base64 string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>String</td>
<td>A base64 encoded string.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The decoded string.</td>
</tr>
</tbody>
</table>

Scoped GlideSystem - beginningOfLastMonth()

Returns the date and time for the beginning of last month in GMT.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT beginning of last month, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped GlideSystem - beginningOfLastWeek()

Returns the date and time for the beginning of last week in GMT.
**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT beginning of last week, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - beginningOfNextMonth()**

Returns the date and time for the beginning of next month in GMT.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT beginning of next month, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - beginningOfNextWeek()**

Returns the date and time for the beginning of next week in GMT.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GMT beginning of next week, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - beginningOfNextYear()**

Returns the date and time for the beginning of next year in GMT.

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
 Scoped GlideSystem - beginningOfThisMonth()

Returns the date and time for the beginning of this month in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT beginning of this month, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

 Scoped GlideSystem - beginningOfThisQuarter()

Returns the date and time for the beginning of this quarter in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT beginning of this quarter, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

 Scoped GlideSystem - beginningOfThisWeek()

Returns the date and time for the beginning of this week in GMT.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - beginningOfThisYear()**

Returns the date and time for the beginning of this year in GMT.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - dateGenerate(String date, String range)**

Generates a date and time for the specified date in GMT.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>String</td>
<td>Format: yyyy-mm-dd</td>
</tr>
<tr>
<td>range</td>
<td>String</td>
<td>Start, end, or a time in the 24 hour format hh:mm:ss.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT beginning of this year, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - daysAgo(Number days)**

Returns the date and time for a specified number of days ago.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A date and time in the format yyyy-mm-dd hh:mm:ss. If range is start, the returned value is yyyy-mm-dd 00:00:00; If range is end the return value is yyyy-mm-dd 23:59:59.</td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>Number</td>
<td>Integer number of days</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

function contractNoticeDue() {
    var gr = new GlideRecord("contract");
    gr.addQuery("u_contract_status", "Active");
    gr.query();
    while (gr.next()) {
        if ((gr.u_termination_date <= gs.daysAgo(-90)) &&
            (gr.u_contract_duration == "Long")) {
            gr.u_contract_status = "In review";
        } else if ((gr.u_termination_date <= gs.daysAgo(-50)) &&
            (gr.u_contract_duration == "Medium")) {
            gr.u_contract_status = "In review";
        } else if ((gr.u_termination_date <= gs.daysAgo(-10)) &&
            (gr.u_contract_duration == "Short")) {
            gr.u_contract_status = "In review";
        }
        gr.update();
    }
}

Scoped GlideSystem - daysAgoEnd(Number days)
Returns the date and time for the end of the day a specified number of days ago.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>Number</td>
<td>Integer number of days</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT end of the day in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped GlideSystem - daysAgoStart(Number days)
Returns the date and time for the beginning of the day a specified number of days ago.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>String</td>
<td>Integer number of days</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT start of the day in the format yyyymm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

```javascript
var gr = new GlideRecord('sysapproval_approver');
gr.addQuery('state', 'requested');
gr.addQuery('sys_updated_on', '<', gs.daysAgoStart(5));
gr.query();
```

Scoped GlideSystem - debug(String message, Object parm1, Object parm2, Object parm3, Object parm4, Object parm5)

Writes a debug message to the system log.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The log message with place holders for any variable arguments.</td>
</tr>
<tr>
<td>param1</td>
<td>Object</td>
<td>(Optional) First variable argument.</td>
</tr>
<tr>
<td>param2</td>
<td>Object</td>
<td>(Optional) Second variable argument.</td>
</tr>
<tr>
<td>param3</td>
<td>Object</td>
<td>(Optional) Third variable argument.</td>
</tr>
<tr>
<td>param4</td>
<td>Object</td>
<td>(Optional) Fourth variable argument.</td>
</tr>
<tr>
<td>param5</td>
<td>Object</td>
<td>(Optional) Fifth variable argument.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
gs.debug("This is a debug message");
var myFirstName = "Abel";
var myLastName = "Tuter";
gs.debug("This is a debug message from {0}.{1}", myFirstName, myLastName);
```

Output:

```
This is a debug message (sys.scripts extended logging)
This is a debug message from Abel.Tuter (sys.scripts extended logging)
```

**Scoped GlideSystem - endOfLastMonth()**

Returns the date and time for the end of last month in GMT.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT end of last month, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - endOfLastWeek()**

Returns the date and time for the end of last week in GMT.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT end of last week, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>
**Scoped GlideSystem - endOfLastYear()**

Returns the date and time for the end of last year in GMT.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - endOfNextMonth()**

Returns the date and time for the end of next month in GMT.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - endOfNextWeek()**

Returns the date and time for the end of next week in GMT.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - endOfNextYear()**

Returns the date and time for the end of next year in GMT.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

 Scoped GlideSystem - endOfThisMonth()

Returns the date and time for the end of this month in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

 Scoped GlideSystem - endOfThisQuarter()

Returns the date and time for the end of this quarter in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

 Scoped GlideSystem - endOfThisWeek()

Returns the date and time for the end of this week in GMT.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

### Scoped GlideSystem - `endOfThisYear()`

Returns the date and time for the end of this year in GMT.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

### Scoped GlideSystem - `error(String message, Object parm1, Object parm2, Object parm3, Object parm4, Object parm5)`

Writes an error message to the system log.

This method accepts up to five variable arguments (varargs) in the message using the Java MessageFormat placeholder replacement pattern.

**Note:** Variables must contain valid values for this method to provide correct output.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The log message with place holders for any variable arguments.</td>
</tr>
<tr>
<td>param1</td>
<td>Object</td>
<td>(Optional) First variable argument.</td>
</tr>
<tr>
<td>param2</td>
<td>Object</td>
<td>(Optional) Second variable argument.</td>
</tr>
<tr>
<td>param3</td>
<td>Object</td>
<td>(Optional) Third variable argument.</td>
</tr>
<tr>
<td>param4</td>
<td>Object</td>
<td>(Optional) Fourth variable argument.</td>
</tr>
<tr>
<td>param5</td>
<td>Object</td>
<td>(Optional) Fifth variable argument.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
gs.error("This is an error message");
var myFirstName = "Abel";
var myLastName = "Tuter";
gs.error("This is an error message from \{0\}.\{1\}", myFirstName, myLastName);
```

Output:

- This is an error message
- This is an error message from Abel.Tuter

Scoped GlideSystem - `eventQueue(String name, Object instance, String parm1, String parm2, String queue)`

Queues an event for the event manager.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the event being queued.</td>
</tr>
<tr>
<td>instance</td>
<td>Object</td>
<td>GlideRecord object, such as 'current'.</td>
</tr>
<tr>
<td>parm1</td>
<td>String</td>
<td>(Optional) Saved with the instance if specified.</td>
</tr>
<tr>
<td>parm2</td>
<td>String</td>
<td>(Optional) Saved with the instance if specified.</td>
</tr>
<tr>
<td>queue</td>
<td>String</td>
<td>Name of the queue.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
Scoped GlideSystem - `eventQueueScheduled(String name, Object instance, String parm1, String parm2, Object expiration)`

Queues an event for the event manager at a specified date and time.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the event being queued.</td>
</tr>
<tr>
<td>instance</td>
<td>Object</td>
<td>A GlideRecord object, such as “current”.</td>
</tr>
<tr>
<td>parm1</td>
<td>String</td>
<td>(Optional) Saved with the instance if specified.</td>
</tr>
<tr>
<td>parm2</td>
<td>String</td>
<td>(Optional) Saved with the instance if specified.</td>
</tr>
<tr>
<td>expiration</td>
<td>Object</td>
<td>Date and time to process this event.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Scoped GlideSystem - `executeNow(GlideRecord job)`

Executes a job for a scoped application.

You can only use this method on a job in the same application as the script calling this method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>job</td>
<td>GlideRecord</td>
<td>The job to be run.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Returns the sysID of the scheduled job. Returns null if the job is global.</td>
</tr>
</tbody>
</table>

Scoped GlideSystem - `generateGUID()`

Generates a GUID that can be used when a unique identifier is required.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A 32-character hexadecimal GUID.</td>
</tr>
</tbody>
</table>

```java
gs.info(personalId); 
```

Output: `af77051f013100e04bffffffff6`

### Scoped GlideSystem - getCallerScopeName()

Gets the caller scope name; returns null if there is no caller.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The caller's scope name, or null if there is no caller.</td>
</tr>
</tbody>
</table>

This example shows two script includes that are set to Accessible from all application scopes. One script include is created within app_scope_a.

```java
var Scopea = Class.create();
Scopea.prototype = {
    initialize: function() {
    },
    callerScope: function() {
        var scopeb = new app_scope_b.Scopeb();
        return scopeb.callerscope();
    },
    type: 'Scopea'
}
```

Output:
This example shows another script include created within app_scope_b.

```javascript
var Scopeb = Class.create();
Scopeb.prototype = {
    initialize: function() {
        this._constructorCallerScope = gs.getCallerScopeName();
    },
    callerscope: function() {
        return gs.getCallerScopeName();
    },
    getConstructorCallerScope: function() {
        return this._constructorCallerScope;
    },
    type: 'Scopeb'
}
```

Output:

This script can be used within scope app_scope_a to get the scope name of the caller—in this case app_scope_b.

```javascript
gs.info(new Scopea().getCallerScopeName());
```

**Scoped GlideSystem - getCssCacheVersionString()**

Gets a string representing the cache version for a CSS file.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The CSS cache version.</td>
</tr>
</tbody>
</table>

```javascript
var verStr = gs.getCssCacheVersionString();
gs.info(verStr);
```

Output: \_d82979516f0171005be8883e6b3ee4cf&theme=

**Scoped GlideSystem - getCurrentApplicationId()**

Gets the ID of the current application as set using the Application Picker.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current application's sys_id, or global in none is set.</td>
</tr>
</tbody>
</table>

```javascript
var currentId = gs.getCurrentApplicationId();
gs.info(currentId);
```

Output: 04936cb16f30b1005be8883e6b3ee4e0

**Scoped GlideSystem - getCurrentScopeName()**

Gets the name of the current scope.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current scope name.</td>
</tr>
</tbody>
</table>

```javascript
var currentScope = gs.getCurrentScopeName();
gs.info(currentScope);
```

**Scoped GlideSystem - getErrorMessages()**

Returns the list of error messages for the session that were added by addErrorMessage().

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>List of error messages</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - getEscapedMessage(String id, Array args)**

Retrieves a message from UI messages that has HTML special characters, and replaces them with escape sequences. For example, `&` becomes `&amp;`.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>ID of the message.</td>
</tr>
<tr>
<td>args</td>
<td>Array</td>
<td>Optional. List of strings or other values defined by java.text.MessageFormat, which allows you to produce language-neutral messages for display to users.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The UI message with HTML special characters replaced with escape sequences.</td>
</tr>
</tbody>
</table>

```javascript
var myMessage = gs.getEscapedMessage("Hello {0}", ["mom"]);
```

**Scoped GlideSystem - getMessage(String id, Array args)**

Retrieves a message from UI messages.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>The ID of the message.</td>
</tr>
<tr>
<td>args</td>
<td>Array</td>
<td>(Optional) a list of strings or other values defined by java.text.MessageFormat, which allows you to produce language-neutral messages for display to users.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The UI message.</td>
</tr>
</tbody>
</table>

```javascript
var my_message = '${gs.getMessage("This is a message.");
alert(my_message);
```

**Scoped GlideSystem - getProperty(String key, Object alt)**

Gets the value of a Glide property. If the property is not found, returns an alternate value.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>String</td>
<td>The key for the property whose value should be returned.</td>
</tr>
<tr>
<td>alt</td>
<td>Object</td>
<td>(Optional) Alternate object to return if the property is not found.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the Glide property, or the alternate object defined above.</td>
</tr>
</tbody>
</table>

```javascript
var attachment_link = gs.getProperty('glide.servlet.uri');
gs.info(attachment_link);
```

Output: https://instance.service-now.com/

**Scoped GlideSystem - getSession()**

Gets a reference to the current Glide session.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A reference for the current session.</td>
</tr>
</tbody>
</table>

```java
if (!gs.hasRole("admin") && !gs.hasRole("user_admin") && gs.getSession().isInteractive()) {
    current.addQuery("active", "true");
}
```

**Scoped GlideSystem - getSessionID()**

Retrieves the GlideSession session ID.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The session ID.</td>
</tr>
</tbody>
</table>

```javascript
var myUserObject = gs.getSessionID();
gs.info(myUserObject);
```

Output:

A0D4E5416F3F21005BE8883E6B3EE4B8

**Scoped GlideSystem - getSessionToken()**

This method is no longer available. Instead, use `gs.getSession().getSessionToken()`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The session token.</td>
</tr>
</tbody>
</table>
Scoped GlideSystem - getTimeZoneName()

Returns the name of the time zone associated with the current user.

This method has been deprecated. Instead, use the `getTimeZoneName()` method in the GlideSession API.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The time zone name.</td>
</tr>
</tbody>
</table>

```
gs.info(gs.getTimeZoneName());
```

Scoped GlideSystem - getUrlOnStack()

Gets the current URI for the session.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The URI.</td>
</tr>
</tbody>
</table>

```
gs.info(gs.getUrlOnStack());
```

Scoped GlideSystem - getUser()

Returns a reference to the scoped GlideUser object for the current user.

See GlideUser for a list of available methods.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### GlideUser

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideUser</td>
<td>Reference to a scoped user object.</td>
</tr>
</tbody>
</table>

```javascript
var myUserObject = gs.getUser();
```

### Scoped GlideSystem - getUserDisplayName()

Gets the display name of the current user.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The name field of the current user. Returns Abel Tuter, as opposed to abel.tuter.</td>
</tr>
</tbody>
</table>

```javascript
gs.info(gs.getUserDisplayName());
```

### Scoped GlideSystem - getUserID()

Gets the sys_id of the current user.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the current user.</td>
</tr>
</tbody>
</table>

```javascript
gs.info(gs.getUserID());
```
**Scoped GlideSystem - getUserName()**

Gets the user name, or user id, of the current user.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The user name of the current user.</td>
</tr>
</tbody>
</table>

```java
gs.info(gs.getUserName());
```

**Scoped GlideSystem - hasRole(Object role)**

Determines if the current user has the specified role.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>Object</td>
<td>The role to check.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user had the role. Returns true for users with the administrator role.</td>
</tr>
</tbody>
</table>

```java
if (!gs.hasRole("admin") && !gs.hasRole("groups_admin") &&
gs.getSession().isInteractive()) {
  var qc = current.addQuery("u_hidden", ",!=" , "true"); //cannot see hidden groups...
  qc.addOrCondition("sys_id", "javascript:getMyGroups()"); //...unless in the hidden group
  gs.info("User has admin and groups admin roles");
} else {
  gs.info("User does not have both admin and groups admin roles");
}
```

**Scoped GlideSystem - hoursAgo(Number hours)**

Returns the date and time for a specified number of hours ago.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hours</td>
<td>Number</td>
<td>Integer number of hours</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

```java
if (current.operation() == 'insert') {
    // If no due date was specified, calculate a default
    if (current.due_date == '') {
        if (current.urgency == '1') {
            // Set due date to 4 hours ahead of current time
            current.due_date = gs.hoursAgo(-4);
        }
        if (current.urgency == '2') {
            // Set due date to 2 days ahead of current time
            current.due_date = gs.daysAgo(-2);
        }
        if (current.urgency == '3') {
            // Set due date to 7 days ahead of current time
            current.due_date = gs.daysAgo(-7);
        }
    }
}
```

**Scoped GlideSystem - hoursAgoEnd(Number hours)**

Returns the date and time for the end of the hour a specified number of hours ago.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hours</td>
<td>Number</td>
<td>Integer number of hours</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - hoursAgoStart(Number hours)**

Returns the date and time for the start of the hour a specified number of hours ago.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hours</td>
<td>Number</td>
<td>Integer number of hours</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

#### Scoped GlideSystem - include(String name)

Provides a safe way to call from the sandbox, allowing only trusted scripts to be included.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name for the script to include.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the include worked.</td>
</tr>
</tbody>
</table>

This example gets the PrototypeServer in the current script context.

```java
gs.include("PrototypeServer");
```

#### Scoped GlideSystem - info(String message, Object parm1, Object parm2, Object parm3, Object parm4, Object parm5)

Writes an info message to the system log.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The log message with placeholders for any variable arguments.</td>
</tr>
<tr>
<td>param1</td>
<td>Object</td>
<td>(Optional) First variable argument.</td>
</tr>
<tr>
<td>param2</td>
<td>Object</td>
<td>(Optional) Second variable argument.</td>
</tr>
<tr>
<td>param3</td>
<td>Object</td>
<td>(Optional) Third variable argument.</td>
</tr>
</tbody>
</table>
Name | Type | Description
--- | --- | ---
param4 | Object | (Optional) Fourth variable argument.
param5 | Object | (Optional) Fifth variable argument.

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
gs.info("This is an info message");
var myFirstName = "Abel";
var myLastName = "Tuter";
gs.info("This is an info message from {0}.{1}", myFirstName, myLastName);
```

Output:

This is an info message
This is an info message from Abel.Tuter

**Scoped GlideSystem - isDebugging()**

Determines if debugging is active for a specific scope.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if either session debugging is active or the log level is set to debug for the specified scope.</td>
</tr>
</tbody>
</table>

```javascript
gs.debug("This is a log message");
var myFirstName = "Abel";
var myLastName = "Tuter";
gs.debug("This is a log message from {0}.{1}", myFirstName, myLastName);
gs.info(gs.isDebugging());
```
**Scoped GlideSystem - isInteractive()**

Checks if the current session is interactive. An example of an interactive session is when a user logs in normally. An example of a non-interactive session is using a SOAP request to retrieve data.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the session is interactive.</td>
</tr>
</tbody>
</table>

```java
if (!gs.hasRole("admin") && gs.getSession().isInteractive()) {
    var qc1 = current.addQuery('u_group',"");
    var gra = new GlideRecord('sys_user_grmember');
    gra.addQuery('user', gs.getUserID());
    gra.query();
    while (gra.next()) {
        qc1.addOrCondition('u_group', gra.group);
    }
}
```

**Scoped GlideSystem - isLoggedIn()**

Determines if the current user is currently logged in.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the current user is logged in.</td>
</tr>
</tbody>
</table>

```java
gs.info(gs.isLoggedIn());
```

**Scoped GlideSystem - isMobile()**

You can determine if a request comes from a mobile device. This method can be used in UI action conditions and business rules.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the request comes from a mobile device; otherwise, false.</td>
</tr>
</tbody>
</table>

```java
if (gs.isMobile())
    gs.info("submitted from mobile UI");
else
    gs.info("NOT submitted from mobile UI");
```

Scoped GlideSystem - monthsAgo(Number months)

Returns the date and time for a specified number of months ago.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>months</td>
<td>Number</td>
<td>Integer number of months</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT on today's date of the specified month, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped GlideSystem - minutesAgoEnd(Number minutes)

Returns the date and time for the end of the minute a specified number of minutes ago.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>minutes</td>
<td>Number</td>
<td>Integer number of minutes</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>
**Scoped GlideSystem - minutesAgoStart(Number minutes)**

Returns the date and time for the start of the minute a specified number of minutes ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>minutes</td>
<td>Number</td>
<td>Integer number of minutes</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - monthsAgoStart(Number months)**

Returns the date and time for the start of the month a specified number of months ago.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>months</td>
<td>Number</td>
<td>Integer number of months</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT start of the month the specified number of months ago, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

**Scoped GlideSystem - nil(Object o)**

Queries an object and returns true if the object is null, undefined, or contains an empty string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Object</td>
<td>The object to be checked.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the object is null, undefined, or contains an empty string; otherwise, returns false.</td>
</tr>
</tbody>
</table>
Scoped GlideSystem - quartersAgoEnd(Number quarters)
Returns the date and time for the last day of the quarter for a specified number of quarters ago.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>quarters</td>
<td>Number</td>
<td>Integer number of quarters</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT end of the quarter that was the specified number of quarters ago, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped GlideSystem - quartersAgoStart(Number quarters)
Returns the date and time for the first day of the quarter for a specified number of quarters ago.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>quarters</td>
<td>Number</td>
<td>Integer number of quarters</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT end of the month that was the specified number of quarters ago, in the format yyyy-mm-dd hh:mm:ss</td>
</tr>
</tbody>
</table>

Scoped GlideSystem - setProperty(String key, String value, String description)
Sets the specified key to the specified value if the property is within the script's scope.

Note: Care should be taken when setting system properties (sys_properties) using this method as it causes a system-wide cache flush. Each flush can cause system degradation while the caches rebuild. If a value must be updated often, it should not be stored as a system property. In general, you should only place values in the sys_properties table that do not frequently change.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>String</td>
<td>The key for the property to be set.</td>
</tr>
</tbody>
</table>
### Name | Type | Description
--- | --- | ---
value | String | The value of the property to be set.
description | String | A description of the property.

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
gs.setProperty("glide.foo","bar","foo");
gs.info(gs.getProperty("glide.foo"));
```

Output: bar

**Scoped GlideSystem - setRedirect(Object o)**

Sets the redirect URI for this transaction, which then determines the next page the user will see.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Object</td>
<td>URI object or URI string to set as the redirect</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
gs.setRedirect("com.glideapp.servicecatalog_cat_item_view.do?sysparm_id=d41ce5bac611227a0167f4bf8109bf70&sysparm_user=" + current.sys_id + "&sysparm_email=" + current.email)
```

**Scoped GlideSystem - tableExists(String name)**

Determines if a database table exists.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the table to check for existence.</td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the table exists. False if the table was not found.</td>
</tr>
</tbody>
</table>

```javascript
gs.info(gs.tableExists('incident'));
```

### Scoped GlideSystem - urlEncode(String url)

Encodes non-ASCII characters, unsafe ASCII characters, and spaces so the returned string can be used on the Internet. Uses UTF-8 encoding. Uses percent (%) encoding.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>String</td>
<td>The string to be encoded.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A string with non-ASCII characters, unsafe ASCII characters, and spaces encoded.</td>
</tr>
</tbody>
</table>

### Scoped GlideSystem - urlDecode(String url)

Replaces UTF-8 encoded characters with ASCII characters.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>String</td>
<td>A string with UTF-8 percent (%) encoded characters.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A string with encoded characters replaced with ASCII characters.</td>
</tr>
</tbody>
</table>

### Scoped GlideSystem - warn(String message, Object parm1, Object parm2, Object parm3, Object parm4, Object parm5)

Writes a warning message to the system log.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The log message with place holders for any variable arguments.</td>
</tr>
<tr>
<td>param1</td>
<td>Object</td>
<td>(Optional) First variable argument.</td>
</tr>
<tr>
<td>param2</td>
<td>Object</td>
<td>(Optional) Second variable argument.</td>
</tr>
<tr>
<td>param3</td>
<td>Object</td>
<td>(Optional) Third variable argument.</td>
</tr>
<tr>
<td>param4</td>
<td>Object</td>
<td>(Optional) Fourth variable argument.</td>
</tr>
<tr>
<td>param5</td>
<td>Object</td>
<td>(Optional) Fifth variable argument.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
gs.warn("This is a warning");
var myFirstName = "Abel";
var myLastName = "Tuter";
gs.warn("This is a warning from {0}.{1}", myFirstName, myLastName);
```

Output:

```
This is a warning
This is a warning from Abel.Tuter
```

### Scoped GlideSystem - xmlToJSON(String xmlString)

 Takes an XML string and returns a JSON object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmlString</td>
<td>String</td>
<td>The XML string to be converted.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>A JSON object representing the XML string. Null if unable to process the XML string.</td>
</tr>
</tbody>
</table>

```javascript
var jsonObject = gs.xmlToJSON(xmlString);
```

Scoped GlideSystem - `yearsAgo(Number years)`

Returns a date and time for a certain number of years ago.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>years</td>
<td>Number</td>
<td>An integer number of years</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT beginning of the year that is the specified number of years ago, in the format <code>yyyy-mm-dd hh:mm:ss</code>.</td>
</tr>
</tbody>
</table>

Scoped GlideSystem - `yesterday()`

Returns yesterday's time (24 hours ago).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>GMT for 24 hours ago, in the format <code>yyyy-mm-dd hh:mm:ss</code></td>
</tr>
</tbody>
</table>

GlideTableHierarchy

The Scoped GlideTableHierarchy API provides methods for handling information about table relationships.
Scoped GlideTableHierarchy - getAllExtensions()
Returns an array of strings containing all tables that extend the current table and includes the current table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of strings containing the tables in the hierarchy that includes the current table.</td>
</tr>
</tbody>
</table>

```javascript
var table = new GlideTableHierarchy("task");
gs.info(table.getAllExtensions());
```

Output: Line breaks added for clarity.
```
task,sc_task,problem_task,change_phase,sc_req_item,kb_submission,
release_phase,problem,ticket,sm_task,hr_task,change_task,change_request,
change_request_imac,incident,release_task,vtb_task,sm_order,hr_case,
sysapproval_group,sc_request
```

Scoped GlideTableHierarchy - getBase()
Returns the parent class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The parent class.</td>
</tr>
</tbody>
</table>

```javascript
var table = new GlideTableHierarchy("cmdb_ci_server");
gs.info(table.getBase());
```

Output:
```
cmdb_ci_computer
```
Scoped GlideTableHierarchy - getHierarchy()

Returns an array of strings containing all classes in the hierarchy of the current table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of strings of the classes in the hierarchy.</td>
</tr>
</tbody>
</table>

```javascript
var table = new GlideTableHierarchy("incident");
gs.info(table.getHierarchy());
```

Output:

```
incident, task
```

Scoped GlideTableHierarchy - getName()

Returns the table's name.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The table's name.</td>
</tr>
</tbody>
</table>

```javascript
var table = new GlideTableHierarchy("incident");
gs.info(table.getName());
```

Output:

```
incident
```

Scoped GlideTableHierarchy - getRoot()

Returns the top level class in the hierarchy.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The root class.</td>
</tr>
</tbody>
</table>

```javascript
var table = new GlideTableHierarchy("cmdb_ci_server");
gs.info(table.getRoot());
```

Output:
```
cmdb_ci
```

---

**Scoped GlideTableHierarchy - getTables()**

Returns an array of strings of the table names in the hierarchy.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of strings containing the names of tables in the hierarchy.</td>
</tr>
</tbody>
</table>

```javascript
var table = new GlideTableHierarchy("incident");
gs.info(table.getTables());
```

Output:
```
incident, task
```

---

**Scoped GlideTableHierarchy - getTableExtensions()**

Returns an array of strings containing all tables that extend the current table.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of strings containing the tables that extend the current table.</td>
</tr>
</tbody>
</table>

```javascript
var table = new GlideTableHierarchy("task");
gs.info(table.getTableExtensions());
```

Output: Line breaks added for clarity

```javascript
sc_task,problem_task,change_phase,sc_req_item,kb_submission,release_phase,problem,ticket,sm_task,hr_task,change_task,change_request,change_request_imac,incident,release_task,vtb_task,sm_order,hr_case,sysapproval_group,sc_request
```

**Scoped GlideTableHierarchy - GlideTableHierarchy(String tableName)**

Instantiates a GlideTableHierarchy object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The name of the table.</td>
</tr>
</tbody>
</table>

```javascript
var table = new GlideTableHierarchy("incident");
gs.info(table.getTables());
```

Output:

```javascript
incident,task
```

**Scoped GlideTableHierarchy - isBaseClass()**

Returns true if this is a base class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the current table has no parent and has extensions.</td>
</tr>
</tbody>
</table>

```javascript
var table = new GlideTableHierarchy("incident");
gs.info(table.isBaseClass());
```

Output:

false

Scoped GlideTableHierarchy - isSoloClass()

Returns true if this table is not in a hierarchy.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the current table has no parent and no extensions.</td>
</tr>
</tbody>
</table>

```javascript
var table = new GlideTableHierarchy("sys_user");
gs.info(table.isSoloClass());
```

Output:

true

Scoped GlideTableHierarchy - hasExtensions()

Returns true if this class has been extended.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
var table = new GlideTableHierarchy("incident");
gs.info(table.hasExtensions());

Output:
false

**GlideTextReader**

Provides the ability to read single lines from an input stream. Because an input stream is used, it is not subject to the 5MB attachment size limit.

**Scoped GlideTextReader - getEncoding()**

Returns the character encoding of the input stream.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Scoped GlideTextReader - GlideTextReader(GlideScriptableInputStream inputStream)**

Creates a scoped GlideTextReader object for the specified input stream.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputStream</td>
<td>GlideScriptableInputStream</td>
<td>The input stream to be read.</td>
</tr>
</tbody>
</table>

**Scoped GlideTextReader - readLine()**

Returns a single line from the input stream and returns a string. Since this is working off of a stream, it is not subject to the 5MB size limit.
The scoped GlideTime class provides methods for performing operations on GlideTime objects, such as instantiating GlideTime objects or working with GlideTime fields.

**Scoped GlideTime - getByFormat(String format)**

Gets the time in the specified format.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>format</td>
<td>String</td>
<td>The time format.</td>
</tr>
</tbody>
</table>

```javascript
var gt = new GlideTime();
gt.setValue('12:00:00');
gs.info(gt.getByFormat("HH:mm"));
```

**Scoped GlideTime - getDisplayValue()**

Gets the time in the current user’s display format and time zone.

```javascript
var is = new GlideSysAttachment().getContentStream(attachmentSysId);
var reader = new GlideTextReader(is);
var ln = ' ';
while((ln = reader.readLine()) != null) {
    gs.info(ln);
}
```
When designing business rules or script includes remember that this method may return values in different formats for different users.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The time in the user's format and time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gt = new GlideTime();
gt.setDisplayValue("12:00:00"); // User Time Zone
gs.info(gt.getDisplayValue()); // User Time Zone
```

**Scoped GlideTime - getDisplayValueInternal()**

Gets the display value in the current user's time zone and the internal format (HH:mm:ss).

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The time value for the GlideTime object in the current user's time zone and the internal time format of HH:mm:ss.</td>
</tr>
</tbody>
</table>

```javascript
var gt = new GlideTime();
gt.setValue("01:00:00"); // Internal Time Zone, UTC
gs.info(gt.getDisplayValueInternal()); // User Time Zone
```

**Scoped GlideTime - getHourLocalTime()**

Returns the hours part of the time using the local time zone.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scoped GlideTime - getHourOfDayLocalTime()

Returns the hours part of the time using the local time zone. The number of hours is based on a 24 hour clock.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The hours using the local time zone. The number of hours is based on a 24 hour clock.</td>
</tr>
</tbody>
</table>

Scoped GlideTime - getHourOfDayUTC()

Returns the hours part of the time using the UTC time zone. The number of hours is based on a 24 hour clock.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The hours using the UTC time zone. The number of hours is based on a 24 hour clock.</td>
</tr>
</tbody>
</table>

Scoped GlideTime - getHourUTC()

Returns the hours part of the time using the UTC time zone. The number of hours is based on a 12 hour clock. Noon and midnight are represented by 0, not 12.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The hours using the UTC time zone. The number of hours is based on a 24 hour clock.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The hours using the UTC time zone. The number of hours is based on a 12 hour clock. Noon and midnight are represented by 0, not 12.</td>
</tr>
</tbody>
</table>

Scoped GlideTime - getMinutesLocalTime()
Returns the number of minutes using the local time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of minutes using the local time zone.</td>
</tr>
</tbody>
</table>

Scoped GlideTime - getMinutesUTC()
Returns the number of minutes in the hour based on the UTC time zone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of minutes in the hour using the UTC time zone.</td>
</tr>
</tbody>
</table>

Scoped GlideTime - getSeconds()
Returns the number of seconds in the current minute.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of seconds in the minute.</td>
</tr>
</tbody>
</table>

**Scoped GlideTime - getValue()**

Gets the time value stored in the database by the GlideTime object in the internal format, HH:mm:ss, and the system time zone.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The time value in the internal format and system time zone.</td>
</tr>
</tbody>
</table>

```javascript
var gt = new GlideTime();
gs.info(gt.getValue()); // Internal Time Zone, UTC
```

**Scoped GlideTime - GlideTime()**

Instantiates a GlideTime object with the current time.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gt = new GlideTime();
gs.info(gt.getDisplayValue());
```

**Scoped GlideTime - GlideTime(Number milliseconds)**

Instantiates a GlideTime object with the specified time.

```javascript
var gt = new GlideTime();
gs.info(gt.getValue());
```
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>Number</td>
<td>The datetime in milliseconds.</td>
</tr>
</tbody>
</table>

```javascript
var gt = new GlideTime(10000);
gs.info(gt.getDisplayValue());
```

### Scoped GlideTime - setDisplayValue(String asDisplayed)

Sets a time value using the current user's display format and time zone.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>asDisplayed</td>
<td>String</td>
<td>The time in the current user's display format and time zone. The parameter must be formatted using the current user's preferred display format, such as HH:mm:ss.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gt = new GlideTime();
gt.setDisplayValue('01:00:00');  // User Time Zone
gs.info(gt.getDisplayValueInternal());  // User Time Zone
```

### Scoped GlideTime - setValue(String o)

Sets the time of the GlideTime object in the internal time zone.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>String</td>
<td>The time in hh:mm:ss format.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gt = new GlideTime();
gt.setValue('01:00:00');  //Internal Time Zone, UTC
gs.info("time is "+ gt.getByFormat('hh:mm:ss'));
```

**Scoped GlideTime - subtract(GlideTime startTime, GlideTime endTime)**

Gets the duration difference between two GlideTime object values.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTime</td>
<td>GlideTime</td>
<td>The start value.</td>
</tr>
<tr>
<td>endTime</td>
<td>GlideTime</td>
<td>The end value.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDuration</td>
<td>The duration between the two values.</td>
</tr>
</tbody>
</table>

```javascript
var gd1 = new GlideTime();
gd1.setDisplayValue("09:00:00");
var gd2 = new GlideTime();
gd2.setDisplayValue("09:10:00");
var dur = GlideDate.subtract(gd1, gd2); //the difference between gdt1 and gdt2
gs.info(dur.getDisplayValue());
```

**GlideTimeline**

The GlideTimeline class provides the core implementation for configuring and displaying a Glide Windowing Toolkit Timeline.

For security, the GlideTimeline has already been instantiated as a single instance variable named `glideTimeline`. All configurations should be made in the client script section of the corresponding schedule page that references this instance variable.

**GlideTimeline - groupByParent(Boolean b)**

Specifies whether or not to group timeline items by their parent. If `true`, this will nest all child items inside their parent. This affects the ordering of display and children will always be listed immediately after their parent. The default value for the `groupByParent` property is `false`. 
If `true`, this will nest all child items inside their parent. This affects the ordering of display and children will always be listed immediately after their parent. The default value for the `groupByParent` property is `false`.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If <code>true</code>, displays Timeline Items grouped by parent.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
glideTimeline.groupByParent(true);
```

**GlideTimeline - registerEvent(String strServerEvent, String strScriptIncludeName)**

Registers the specified Timeline server event. The `strServerEvent` must be one of the allowed events for registration to work correctly. When the event occurs, the GlideTimeline sends a request to the server and processes the event as handled inside the `strScriptIncludeName` class.

The `strServerEvent` must be one of the allowed events for registration to work correctly. When the event occurs, the GlideTimeline sends a request to the server and processes the event as handled inside the `strScriptIncludeName` class.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strServerEvent</td>
<td>String</td>
<td>Specifies one of the following case-sensitive events:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>getItems</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>elementMoveX</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>elementMoveY</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>elementMoveXY</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>elementSuccessor</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>elementTimeAdjustStart</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>elementTimeAdjustEnd</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>inputBox</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>itemMove</code></td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strScriptIncludeName</td>
<td>String</td>
<td>Specifies the name of the class to receive the strServerEvent. This class must be defined in a script include that extends AbstractTimelineSchedulePage.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
glideTimeline.registerEvent("getItems", "TimelineGanttSchedulePage");
```

**GlideTimeline - setAutoRefresh(Number intSeconds)**

Specifies the number of seconds to wait before performing an auto refresh of the data on the timeline. Setting the number of seconds to 0 will turn auto refresh off. By default, auto refresh is disabled. If **intSeconds** is greater than 0 and less than the minimum allowed time in seconds (10), it will be set to 10 seconds.

Setting the number of seconds to 0 will turn auto refresh off. By default, auto refresh is disabled. If **intSeconds** is greater than 0 and less than the minimum allowed time in seconds (10), it will be set to 10 seconds.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intSeconds</td>
<td>Number</td>
<td>An integer specifying the time in seconds between auto-refreshing.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
glideTimeline.setAutoRefresh(15); // Sets the interval for auto-refreshing to 15 seconds.
```
GlideTimeline - setDefaultPointIconClass(String icon_class)

Specifies the default icon class to use for Timeline Spans with zero duration if no icon class was explicitly specified in the properties of the Timeline Span returned from the server. The default value for the setDefaultPointIconClass property is milestone.

The default value for the setDefaultPointIconClass property is milestone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>icon_class</td>
<td>String</td>
<td>String that specifies one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· milestone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· blue_square</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· sepia_square</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· green_square</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· red_square</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· black_square</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· blue_circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· sepia_circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· green_circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· red_circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· black_circle</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

glideTimeline.setDefaultPointIconClass('blue_circle');

GlideTimeline - setExtraAjaxParam(String strName, String strValue)

Allows setting of additional parameters in the client script to be made available to the corresponding Script Include events by using the getParameter() method. URI parameters that are prefixed with sysparm_timeline_ will automatically be included in all server side AJAX calls.

URI parameters that are prefixed with sysparm_timeline_ will automatically be included in all server side AJAX calls.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strName</td>
<td>String</td>
<td>The URI parameter name.</td>
</tr>
<tr>
<td>strValue</td>
<td>String</td>
<td>The value of strName.</td>
</tr>
</tbody>
</table>
GlideTimeline - setInitialViewRange(String objStartDate, String objEndDate)

Specifies the initial viewable range for the timeline. The format of the start and end dates must be in the default timestamp format: yyyy-MM-dd HH:mm:ss. The default range is the range that specifies the earliest Timeline Span point to the end of the latest Timeline Span. If the initialViewRange property is specified, it will override the default range.

The format of the start and end dates must be in the default timestamp format: yyyy-MM-dd HH:mm:ss. The default range is the range that specifies the earliest Timeline Span point to the end of the latest Timeline Span. If the initialViewRange property is specified, it will override the default range.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objStartDate</td>
<td>String</td>
<td>The start time of the view range in format: yyyy-MM-dd HH:mm:ss.</td>
</tr>
<tr>
<td>objEndDate</td>
<td>String</td>
<td>The end time of the view range in format: yyyy-MM-dd HH:mm:ss.</td>
</tr>
</tbody>
</table>

GlideTimeline - setInitialViewRange(Number objStartDate, Number objEndDate)

Specifies the initial viewable range for the timeline. The default range is the range that specifies the earliest Timeline Span point to the end of the latest Timeline Span. If the initialViewRange property is specified, it will override the default range.
The default range is the range that specifies the earliest Timeline Span point to the end of the latest Timeline Span. If the **initialViewRange** property is specified, it will override the default range.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objStartDate</td>
<td>Number</td>
<td>The start time of the view range in milliseconds.</td>
</tr>
<tr>
<td>objEndDate</td>
<td>Number</td>
<td>The end time in milliseconds.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
// Sets the initial range to begin on June 20th, 2010 at 8:00 AM and end on June 28th, 2010 at 2:00 PM UTC time.
glideTimeline.setInitialViewRange(1277046000000, 1277647200000);
```

### GlideTimeline - setInitialViewRange(String objStartDate, Number objEndDate)

Specifies the initial viewable range for the timeline. The default range is the range that specifies the earliest Timeline Span point to the end of the latest Timeline Span. If the **initialViewRange** property is specified, it will override the default range.

The default range is the range that specifies the earliest Timeline Span point to the end of the latest Timeline Span. If the **initialViewRange** property is specified, it will override the default range.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objStartDate</td>
<td>String</td>
<td>The start time of the view range in format: yyyy-MM-dd HH:mm:ss.</td>
</tr>
<tr>
<td>objEndDate</td>
<td>Number</td>
<td>The end time in milliseconds.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
// Sets the initial range to begin on June 20th, 2010 at 8:00 AM and end on June 28th, 2010 at 2:00 PM UTC time.
```
GlideTimeline - setInitialViewRange(Number objStartDate, String objEndDate)

Specifies the initial viewable range for the timeline. The default range is the range that specifies the earliest Timeline Span point to the end of the latest Timeline Span. If the `initialViewRange` property is specified, it will override the default range.

The default range is the range that specifies the earliest Timeline Span point to the end of the latest Timeline Span. If the `initialViewRange` property is specified, it will override the default range.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objStartDate</td>
<td>Number</td>
<td>The start time of the view range in milliseconds.</td>
</tr>
<tr>
<td>objEndDate</td>
<td>String</td>
<td>The end time of the view range in format: yyyy-MM-dd HH:mm:ss.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```java
// Sets the initial range to begin on June 20th, 2010 at 8:00 AM and end on June 28th, 2010 at 2:00 PM UTC time.
glideTimeline.setInitialViewRange(1277046000000, "2010-06-28 14:00:00");
```

GlideTimeline - setReadOnly(Boolean b)

Enables or disables all timeline event interaction. If enabled, event interaction is determined from the corresponding attributes specified by each Timeline Item. The default value for the `readOnly` property is `false`.

If enabled, event interaction is determined from the corresponding attributes specified by each Timeline Item. The default value for the `readOnly` property is `false`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If <code>true</code>, marks the entire timeline as read-only (non-interactive).</td>
</tr>
</tbody>
</table>
GlideTimeline - showDependencyLines(Boolean b)

Specifies whether or not to show dependency lines between Timeline Spans. This method applies only if the set of Timeline Items returned from the server includes dependency relationships. The default value for the showDependencyLines property is false.

This method applies only if the set of Timeline Items returned from the server includes dependency relationships. The default value for the showDependencyLines property is false.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If true, displays dependency lines on the timeline; otherwise, does not.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideTimeline - showGridLines(Boolean bShowGridlines, Number amount)

Specifies whether or not to show grid lines for each row of data on the timeline. By default, grid lines are enabled.

By default, grid lines are enabled.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bShowGridlines</td>
<td>Boolean</td>
<td>If true, timeline shows grid lines; otherwise, does not display grid lines.</td>
</tr>
<tr>
<td>amount</td>
<td>Number</td>
<td>amount</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>ReturnValue</td>
</tr>
</tbody>
</table>

```java
glideTimeline.showGridLines(false); // Disables grid lines.
```

**GlideTimeline - showLeftPane(Boolean b)**

Specifies whether or not to show the left hand pane in the timeline. The default value for the `leftPane` property is `true`.

The default value for the `leftPane` property is `true`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If <code>true</code>, the timeline shows the left pane; otherwise, the left pane is not be displayed.</td>
</tr>
</tbody>
</table>

```java
glideTimeline.showLeftPaneAsTree(false);
```

**GlideTimeline - showLeftPaneAsTree(Boolean b)**

Specifies how to show child items in the left pane of the timeline. Child items are displayed either as nested, indented nodes with expand and collapse capability or on a single indent level. The default value for the `showLeftPaneAsTree` property is `false`.

The default value for the `showLeftPaneAsTree` property is `false`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If <code>true</code>, child item nodes are shown as indented with expand/collapse capability; otherwise, all left pane items are displayed at a single indent level.</td>
</tr>
</tbody>
</table>
GlideTimeline - showLeftPaneInputBox(Boolean b, String strDefaultValue)

Specifies whether or not to show the text input box at the bottom of the left pane with a default value as specified by `strDefaultValue`. If the left pane is disabled via `showLeftPane()`, the input box will not be visible. The default value for the `showLeftPanelInputBox` property is `false`.

If the left pane is disabled via `showLeftPane()`, the input box will not be visible. The default value for the `showLeftPanelInputBox` property is `false`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If <code>true</code>, show the left pane input box.</td>
</tr>
<tr>
<td>strDefaultValue</td>
<td>String</td>
<td>The default value to display in the input box.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
glideTimeline.showLeftPaneAsTree(true);
```

```
GlideTimeline - showSummaryPane(Boolean b)

Specifies whether or not to show the summary pane at the bottom of the timeline. The default value for the `showSummaryPane` property is `true`.

The default value for the `showSummaryPane` property is `true`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If <code>true</code>, the timeline includes the summary pane; otherwise, the summary pane is not displayed.</td>
</tr>
</tbody>
</table>

```
glideTimeline.showLeftPanelInputBox(true, 'Add a new task ...');
```
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

glideTimeline.showSummaryPane(false);

**GlideTimeline - showTimelineText(Boolean b)**

Specifies whether or not to show the timeline text underneath each Timeline Span in the primary timeline pane. The default value for the showTimelineText property is **false**.

The default value for the showTimelineText property is **false**.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If true, displays descriptive text underneath each Timeline Span; otherwise, no text is displayed underneath each Timeline Span.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

glideTimeline.showTimelineText(true);

**GlideTimeline - snapVertScrollingIntoRows(Boolean b)**

Specifies whether or not the vertical movement of timeline span objects (if appropriately registered to perform this event) should snap adjust into the closest row. By default this value is enabled.

By default this value is enabled.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If true, snaps vertical movement of timeline span objects into rows; otherwise, items move exactly with respect to the mouse.</td>
</tr>
</tbody>
</table>
GlideTimeline - sortByLeftLabelText(Boolean b)

Specifies whether or not to group items by their left label text. The default value for the `sortByLeftLabelText` property is `false`.

The default value for the `sortByLeftLabelText` property is `false`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If <code>true</code>, sort Timeline Items alphabetically by the text specified in each item's left label.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideTimeline - sortByStartDate(Boolean b)

Specifies whether or not to sort the list of Timeline Items returned by the earliest start date of an item’s Timeline Span objects. If `groupByParent()` is set `true`, items are sorted after grouping has occurred. The default value for the `sortByStartDate` property is `false`.

If `groupByParent()` is set `true`, items are sorted after grouping has occurred. The default value for the `sortByStartDate` property is `false`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If <code>true</code>, sort Timeline Items chronologically starting with their earliest start date.</td>
</tr>
</tbody>
</table>
## GlideTimeline - sortByTimelineLabelText(Boolean b)

Specifies whether or not to sort the list of Timeline Items returned in alphabetical order according to the text that was specified to show in the Timeline Pane.

**Note:** This sort order still applies even if the timeline text has been set `false` via the `showTimelineText()` method. Additionally, if `groupByParent()` is set `true`, items will be sorted appropriately after grouping has occurred.

The default value for the `sortByTimelineLabelText` property is `false`.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>If true, sorts Timeline Items alphabetically by the text specified in each item's timeline span text.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
glideTimeline.sortByStartDate(true);
```

```java
GlideTimelineItem
```

GlideTimelineItem extends the abstract `ScheduleItem` class to define additional properties that are specific to the time line.

A time line item is essentially any item that is displayed in a singular row across the time line. A `GlideTimelineItem` has zero or more associated spans (`TimelineSpan` objects).

### GlideTimelineItem - createTimelineSpan(String tableName)

Creates a new `TimelineSpan` object associated with the current instance object.

If no other `TimelineSpan` objects exist, the newly created object will share the same `sys_id` as current instance object. Otherwise, a randomly generated GUID will be used.

```java
gslideTimeline.sortByTimelineLabelText(true);
```
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The name of the table associated with current object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The newly-created span object instance.</td>
</tr>
</tbody>
</table>

**GlideTimelineItem - createTimelineSpan(String tableName, String sys_id)**

Creates a new `TimelineSpan` object associated with the current instance object using the specified table and `sysId`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The name of the table associated with current object.</td>
</tr>
<tr>
<td>sys_id</td>
<td>String</td>
<td>The sys ID for the object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The newly-created span object instance.</td>
</tr>
</tbody>
</table>

**GlideTimelineItem - getImage()**

Returns a string specifying the name of the image file associated with the current `GlideTimelineItem`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The name of the image file associated with the current <code>GlideTimelineItem</code>. If no image is associated with the current item, an empty string (&quot;&quot;) is returned.</td>
</tr>
</tbody>
</table>
GlideTimelineItem - getIsDroppable()

Indicates whether or not the current instance object should be allowed as a "drop zone" when moving timeline elements vertically.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if droppable; false otherwise.</td>
</tr>
</tbody>
</table>

GlideTimelineItem - getLeftLabelText()

Returns the text to be displayed in the left pane (if enabled).

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the text to be displayed in the left pane.</td>
</tr>
</tbody>
</table>

GlideTimelineItem - getParent()

Returns the unique sysId of the current GlideTimelineItem's parent object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The unique sysId of the current GlideTimelineItem's parent object. If the parent does not exist, this will return an empty string (&quot;&quot;).</td>
</tr>
</tbody>
</table>
GlideTimelineItem - getTimelineSpans()

Returns all the TimelineSpan objects associated with the current instance in an ArrayList.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Array</td>
<td>The list of TimelineSpan objects associated with the current instance.</td>
</tr>
</tbody>
</table>

GlideTimelineItem - GlideTimelineItem(String tableName)

Create a “dummy” GlideTimelineItem object.

This is useful for creating rows that do not allow any YMoving into; however, contain nested children (e.g. The top-level "Users" row in the Group Resource Timeline). The sys_id needs to be unique for DOM level functions to parse correctly. By default this object will not be “droppable” because a table name was not specified.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The name of the table associated with current object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

GlideTimelineItem - GlideTimelineItem(String tableName, String sys_id)

Constructor that sets the required table and sys_id properties.

The rest of this object’s properties should be set by the caller. By default, this object instance is “droppable” since a table name is specified.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The name of the table associated with current object.</td>
</tr>
<tr>
<td>sys_id</td>
<td>String</td>
<td>The sys ID for the object.</td>
</tr>
</tbody>
</table>
## GlideTimelineItem - isTextBold()

**Description**
Indicates if the left pane text is set to be displayed using a bold style.

**Parameters**
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the text should be bolded; otherwise false.</td>
</tr>
</tbody>
</table>

## GlideTimelineItem - setImage(String strImageName)

**Description**
Sets the name of the image file (including it's path) to use as the icon for the item in the left pane.

**Parameters**
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strImageName</td>
<td>String</td>
<td>The name of the image, including its path.</td>
</tr>
</tbody>
</table>

**Returns**
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

## GlideTimelineItem - setIsDraggable(Boolean b)

**Description**
Sets whether or not the current instance object can be clicked and dragged into another GlideTimelineItem.

**Parameters**
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>True if item should be able to be moved with click and drag. Otherwise, false.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**GlideTimelineItem - setLeftLabelText(String strText)**

Specifies the text to display in the left pane for this item.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strText</td>
<td>String</td>
<td>The text to display in the left pane for this item.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideTimelineItem - setParent(String sysId)**

Sets the parent of the current GlideTimelineItem.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysId</td>
<td>String</td>
<td>The sysId of the GlideTimelineItem that should become the parent of the current GlideTimelineItem.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideTimelineItem - setTextBold(Boolean b)**

Specifies whether or not to bold the text style of the item in the left pane.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Boolean</td>
<td>True if text in left pane should be bolded; otherwise false.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**GlideUICompatibility**

The scoped GlideUICompatibility class provides the ability for scoped applications to define their own minimum browser versions. This is done by creating system properties for the scoped application.

You create the properties using the `sys_properties` list and assign a version number. When you do this from the scoped application, the `<scope-name>` prefix is automatically added to the property name. The scoped application UI compatibility properties are:

- `<scope-name>.ui.ie_minimum`
- `<scope-name>.ui.chrome_minimum`
- `<scope-name>.ui.firefox_minimum`
- `<scope-name>.ui.safari_major_version_minimum`

You can then use the scoped GlideUICompatibility class to determine if the current browser is supported.

**Scoped GlideUICompatibility - getCompatibility()**

Returns the terms "block" or "allow" based upon the browser version.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Either block or allow</td>
</tr>
</tbody>
</table>

```java
UICompatibility = new GlideUICompatibility(gs.getCurrentScopeName());
var blockOrAllow = UICompatibility.getCompatibility();
gs.info(blockOrAllow);
```

Output: allow

**Scoped GlideUICompatibility - GlideUICompatibility(String scopeName)**

Creates a GlideUICompatibility object.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scopeName</td>
<td>String</td>
<td>The application’s scope name</td>
</tr>
</tbody>
</table>

**Scoped GlideUICompatibility - isBlocked()**

Determines if the browser is not supported.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the browser is not supported.</td>
</tr>
</tbody>
</table>

```java
UICompatibility = new GlideUICompatibility(gs.getCurrentScopeName());
var blocked = UICompatibility.isBlocked();
gs.info(blocked);
```

Output: false

**GlideURI**

The GlideURI class is a utility class for handling the URI parameter. The GlideURI class is available in scoped and global scripts.

**GlideURI - get(String name)**

Returns the specified parameter.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The parameter name.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The URI for the specified parameter.</td>
</tr>
</tbody>
</table>

```java
var gURI = new GlideURI();
```
gURI.set('sysparm_query', 'priority=2^active=true');
var fileString = gURI.get('sysparm_query');
gs.info(fileString);

Output:
priority=2^active=true

GlideURI - getFileFromPath()

Returns the file name portion of the URI.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The file name portion of the URI.</td>
</tr>
</tbody>
</table>

var gURI = new GlideURI();
var fileString = gURI.getFileFromPath();
gs.info(fileString);

GlideURI - GlideURI()

Instantiates a GlideURI object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GlideURI - set(String name, String value)

Sets the specified parameter to the specified value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The parameter name.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value.</td>
</tr>
</tbody>
</table>
ServiceNow Kingston Now Platform Custom Business Applications

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var gURI = new GlideURI();
gURI.set('sysparm_query', 'priority=2^active=true');
var fileString = gURI.get('sysparm_query');
gs.info(fileString);
```

Output:
```
priority=2^active=true
```

**GlideURI** - **toString(String path)**

Reconstructs the URI string and performs the proper URL encoding by converting non-valid characters to their URL code. For example, converting & to %26.

Parameters set with the set method are encoded with the URI as well.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>String</td>
<td>The base portion of the system URL to which the URI is appended.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The URL.</td>
</tr>
</tbody>
</table>

```javascript
fileString = gURI.toString('https://<your instance>.service-now.com/navpage.do');
```

**GlideURLV3**

Provides methods for manipulating a URI.

The GlideURLV3 API can be used in client-side scripts using ListV2 and ListV3 APIs.

**Note:** This API is not supported by Service Portal.

**GlideURLV3 - addParam(String name, String value)**

Adds a query string name-value pair to the URL.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the query string parameter.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>Query string value.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GlideURL</td>
</tr>
</tbody>
</table>

```javascript
var gu = new GlideURL('incident.do');
var url = gu.addParam('sys_id', '-1');
```

**GlideURLV3 - getURL(Object additionalParams)**

Get the entire context path and query string parameters as a single URI.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>additionalParams</td>
<td>Object</td>
<td>A name-value pair object that contains parameters that are added to this URL request only. These additional parameters are not saved to the GlideURL object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The GlideURL with the specified additional parameters added to the end.</td>
</tr>
</tbody>
</table>

**GlideURLV3 - GlideURL(String contextPath)**

Creates an instance of the GlideURL class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextPath</td>
<td>String</td>
<td>A relative path for the URL.</td>
</tr>
</tbody>
</table>

**GlideURLV3 - refresh()**

Reloads the current page URL.
This refreshes the current page URL, not the URL set in the object.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### GlideUser

The GlideUser API provides access to information about the current user and current user roles. Using the GlideUser API avoids the need to use the slower GlideRecord queries to get user information.

The GlideUser methods and properties are accessed through a global object (g_user) that is only available in client scripts. GlideUser

- contains name and role information about the current user.
- is typically used in client scripts and UI policies but is also found in UI actions that run on the client.
- cannot be used in business rules or UI actions that run on the server.
- avoids the need for GlideRecord queries to get user information.

Session information about the current user and current user roles is contained in the client (web browser). All GlideUser methods except `getClientData()` access the session information that is available by default. The `getClientData()` method requires setup on the server and use of `putClientData()` to make session information available.

#### GlideUser - firstName

Returns the current user’s first name.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>firstName</td>
<td>String</td>
<td>Current user’s first name.</td>
</tr>
</tbody>
</table>

```javascript
alert('first name = ' + g_user.firstName);
```

#### GlideUser - getClientData(string key)

Returns a session client value previously set with `putClientData()`.

Session client data is a set of named strings that may be setup on the server (using `putClientData()`) that then may be used by client scripts (using `getClientData()`). Can be
used during form load time to get information that the client script needs to make decisions about the form, for example, which fields should be visible.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>String</td>
<td>Name of the client data to retrieve.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Value of the client data.</td>
</tr>
</tbody>
</table>

```javascript
var loginLanguage = g_user.getClientData("loginlanguage");
```

**GlideUser - getFullName()**

Returns the first and last name of the current user.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The current user's full name.</td>
</tr>
</tbody>
</table>

```javascript
var formalName = g_user.getFullName();
```

**GlideUser - hasRole(String role, Boolean includeDefaults)**

Returns true if the current user has the specified role or the admin role.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>String</td>
<td>Role to check.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>includeDefaults</td>
<td>Boolean</td>
<td>(Optional) Flag that indicates whether to include default roles, such as snc_internal and snc_external, in the request. For additional information on roles, see Explicit roles. Default: false</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the current user has the specified role or the admin role; otherwise returns false.</td>
</tr>
</tbody>
</table>

```javascript
var isInternal = g_user.hasRole('snc_internal', true);
```

```javascript
var isItil = g_user.hasRole('itil');
```

**GlideUser - hasRoleExactly(String role, Boolean includeDefaults)**

Returns true only if the current user has the specified role.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>String</td>
<td>Role to check.</td>
</tr>
<tr>
<td>includeDefaults</td>
<td>Boolean</td>
<td>(Optional) Flag that indicates whether to include default roles, such as snc_internal and snc_external, in the request. For additional information on roles, see Explicit roles. Default: false</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the current user has the specified role.</td>
</tr>
</tbody>
</table>

```javascript
var isInternal = g_user.hasRoleExactly('snc_internal', true);
```

```javascript
var isItil = g_user.hasRoleExactly('itil');
```

### GlideUser - hasRoleFromList(String roles, Boolean includeDefaults)

Returns true if the current user has at least one of the specified roles or has the admin role.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roles</td>
<td>String</td>
<td>Comma-separated list of roles to check.</td>
</tr>
<tr>
<td>includeDefaults</td>
<td>Boolean</td>
<td>(Optional) Flag that indicates whether to include default roles, such as snc_internal and snc_external, in the request. For additional information on roles, see <a href="#">Explicit roles</a>. Default: false</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the current user has a role in the list or the admin role.</td>
</tr>
</tbody>
</table>

```javascript
var isOK = g_user.hasRoleFromList("itil, maint");
```

```javascript
var isOK = g_user.hasRoleFromList("itil, maint, snc_internal", true);
```
**GlideUser - hasRoles(Boolean includeDefaults)**

Returns true if the current user has any role.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeDefaults</td>
<td>Boolean</td>
<td>(Optional) Flag that indicates whether to include default roles, such as snc_internal and snc_external, in the request. For additional information on roles, see <a href="#">Explicit roles</a>. Default: false</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the current user has at least one role.</td>
</tr>
</tbody>
</table>

```javascript
var yesRole = g_user.hasRoles();

var yesRole = g_user.hasRoles(true);
```

**GlideUser - lastName**

The current user's last name.

**Field**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lastName</td>
<td>String</td>
<td>Current user's last name.</td>
</tr>
</tbody>
</table>

```javascript
alert('last name = ' + g_user.lastName);
```

**GlideUser - userID**

Returns the sys_id of the current user.
ServiceNow, Kingston, Now Platform Custom Business Applications

<table>
<thead>
<tr>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>userID</td>
</tr>
</tbody>
</table>

```
var userID = g_user.userID;
alert('Current user ID = ' + userID);
```

**GlideUser - userName**

This property is the current user's username, for example gsmith02. It is not the user's name, for example George Smith.

```
var userName = g_user.userName;
alert('Current user = ' + userName);
```

**GlideUser**

The scoped GlideUser API provides access to information about the current user and current user roles. Using the scoped GlideUser API avoids the need to use the slower GlideRecord queries to get user information.

**Scoped GlideUser - getCompanyID()**

Returns the current user's company sys_id.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Company sys_id</td>
</tr>
</tbody>
</table>

```
var currentUser = gs.getUser();
gs.info(currentUser.getCompanyID());
```
Scoped GlideUser - getDisplayname()
Returns the current user's display name.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User's display name</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getDisplayName());
```

Scoped GlideUser - getDomainID()
Returns the identifier of the user's current session domain.
The identifier that is returned depends on the domain type and the instantiation of that domain.
- If the user is configured in the global domain, and does not use the domain picker to switch domains, the method returns null.
- If the user uses the domain picker to switch to the global domain, the method returns the string "global".
- For all other domains, the method returns the sys_id of that domain.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Domain identifier</td>
</tr>
</tbody>
</table>

```javascript
var domain = new GlideRecord('domain');
domain.get(gs.getUser().getDomainID());
gs.info(domain.name);
```

Scoped GlideUser - getEmail()
Returns the user's email address.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User's email address</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getEmail());
```

Scoped GlideUser - getFirstName()

Returns the user's first name.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User's first name</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getFirstName());
```

Scoped GlideUser - getID()

Gets the sys_id of the current user.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getID());
```
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User’s sys_id</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getID());
```

 Scoped GlideUser - getLastName()

Returns the user’s last name.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User’s last name</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getLastName());
```

 Scoped GlideUser - getName()

Returns the user ID, or login name, of the current user.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User ID</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getName());
```
**Scoped GlideUser - getPreference(String name)**

Gets the specified user preference value for the current user.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the preference.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The preference value.</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
currentUser.savePreference('-myPref','red');
gs.info(currentUser.getPreference('-myPref'));
```

**Scoped GlideUser - getRoles()**

Returns a list of roles that includes explicitly granted roles, inherited roles, and roles acquired by group membership.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>List of all roles available to the user</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getRoles());
```

**Scoped GlideUser - getUserRoles()**

Returns the list of roles explicitly granted to the user.

Unlike the `getRoles()` method, this method does not return roles the user inherits or roles acquired from group membership.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>List of roles explicitly assigned to the user</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getUserRoles());
```

**Scoped GlideUser - hasRole(String role)**

Determines if the current user has the specified role.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>String</td>
<td>Role to check</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user has the role.</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.hasRole('admin'));
```

**Scoped GlideUser - isMemberOf(String group)**

Determines if the current user is a member of the specified group.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>String</td>
<td>Group to check</td>
</tr>
</tbody>
</table>
var currentUser = gs.getUser();
gs.info(currentUser.isMemberOf('-' + 'Capacity Mgmt'));

### Scoped GlideUser - `savePreference(String name, String value)`

Saves a user preference value to the database.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The preference to save.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The preference value.</td>
</tr>
</tbody>
</table>

**Returns**

| Type   | Description | |
|--------|-------------| |
| void   |             | |

var currentUser = gs.getUser();
currentUser.savePreference('myPref','red');
gs.info(currentUser.getPreference('myPref'));

### GlideUser

The GlideUser API provides access to information about the current user and current user roles. Using the GlideUser API avoids the need to use the slower GlideRecord queries to obtain user information.

### GlideUser - `getCompanyID()`

Returns the current user’s company `sys_id`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Company sys_id</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getCompanyID());
```

**Scoped equivalent**

To use the `getCompanyID()` method in a scoped application, use the corresponding scoped method: `getCompanyID()`.

**GlideUser - getDisplayName()**

Returns the current user's display name.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User's display name</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getDisplayName());
```

**Scoped equivalent**

To use the `getDisplayName()` method in a scoped application, use the corresponding scoped method: `getDisplayName()`.

**GlideUser - getDomainDisplayValue()**

Returns the display value of the user's session domain.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The display value of the user’s session domain.</td>
</tr>
</tbody>
</table>

Optional example explanation

```
var currentUser = gs.getUser();
gs.info(currentUser.getDomainDisplayValue());
```

Scoped equivalent

There is no workaround for scoped applications.

**GlideUser - getEmail()**

Returns the user’s email address.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User’s email address</td>
</tr>
</tbody>
</table>

```
var currentUser = gs.getUser();
gs.info(currentUser.getEmail());
```

Scoped equivalent

To use the `getEmail()` method in a scoped application, use the corresponding scoped method: `getEmail()`.

**GlideUser - getFirstName()**

Returns the user’s first name.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User's first name</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getFirstName());
```

#### Scoped equivalent

To use the `getFirstName()` method in a scoped application, use the corresponding scoped method: `getFirstName()`.

---

### GlideUser - `getID`

Returns the **sys_id** of the current user.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User's <strong>sys_id</strong></td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getID());
```

#### Scoped equivalent

To use the `getID()` method in a scoped application, use the corresponding scoped method: `getID()`.

---

### GlideUser - `getLastName()`

Returns the user's last name.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User's last name</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getLastName());
```

### Scoped equivalent

To use the `getLastName()` method in a scoped application, use the corresponding scoped method: `getLastName()`.

### GlideUser - getMyGroups()

Returns an iterator containing the list of all groups to which the user belongs. Only active groups are returned.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterator</td>
<td>A list of sys_ids for the active groups to which the user belongs.</td>
</tr>
</tbody>
</table>

An example

```javascript
var groupsArray = gs.getUser().getMyGroups().toArray();
gs.info(groupsArray[0]);
```

Output: cfcbad03d711110050f5edcb9e61038f

### Scoped equivalent

There is no scoped equivalent for this method.

### GlideUser - getName()

Returns the user ID, or login name, of the current user.
**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>User ID</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getName());
```

**Scoped equivalent**

To use the `getName()` method in a scoped application, use the corresponding scoped method: `getName0`.

**GlideUser - getRoles()**

Returns a list of roles that includes explicitly granted roles, inherited roles, and roles acquired by group membership.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>List of all roles available to the user</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getRoles());
```

**Scoped equivalent**

To use the `getRoles()` method in a scoped application, use the corresponding scoped method: `getRoles()`.  

**GlideUser - getUserByID (String id)**

Returns the user object associated with the passed-in user ID (sys_id in sys_user) or user_name.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Unique ID (sys_id) or user_name of the desired user record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>User object associated with the specified sys_id or user_name.</td>
</tr>
</tbody>
</table>

Example using user name (user_name).

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getFirstName()); // print the first name of the logged in user
var newUser = currentUser.getUserByID('abel.tuter'); // fetch a different user using the user_name field
gs.info(newUser.getFirstName()); // print the first name of the Abel Tuter user
```

Example using user ID (sys_id).

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getFirstName()); // print the first name of the logged in user
var newUser = currentUser.getUserByID('62826bf03710200044e0bfc8bcbe5df1'); // fetch Abel Tuter user using sys_id from sys_user record
gs.info(newUser.getFirstName()); // print the first name of the Abel Tuter user
```

GlideUser - getUserRoles()

Returns the list of roles explicitly granted to the user.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>List of roles explicitly assigned to the user</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.getUserRoles());
```

### Scoped equivalent

To use the `getUserRoles()` method in a scoped application, use the corresponding scoped method: `getUserRoles()`.

#### GlideUser - hasRole(String role)

Determines if the current user has the specified role.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>String</td>
<td>Role to check</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user has the role.</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.hasRole('admin'));
```

### Scoped equivalent

To use the `hasRole()` method in a scoped application, use the corresponding scoped method: `hasRole()`.

#### GlideUser - isMemberOf(String group)

Determines if the current user is a member of the specified group.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>String</td>
<td>Group to check</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the user is a member of the group.</td>
</tr>
</tbody>
</table>

```javascript
var currentUser = gs.getUser();
gs.info(currentUser.isMemberOf(-'Capacity Mgmt'));
```

Scoped equivalent

To use the `isMemberOf()` method in a scoped application, use the corresponding scoped method: `isMemberOf()`.

HistoryWalker -

The HistoryWalker API uses the audit/history tables to generate a historical version of an existing record. It supports the ability to return a GlideRecord to a previous update count (walked GlideRecord) with the appropriate GlideElements populated. After the walked GlideRecord is retrieved, the API provides the ability to move forward and backward the update numbers navigating through its historical updates.

The associated methods can be used in scoped and global applications. To use this class in scoped and global applications, use the `sn_hw` namespace identifier. The History Walker plugin (com.glide.history_walker) that is enabled by default is required to access the `HistoryWalker` API.

The HistoryWalker API provides two ways to retrieve the audit data:

- Using History Set: A History Set entry is created (if not available or not up to date) from the data in the `sys_audit` table for the record that you are going to walk through. The History Set table contains records (History Lines) with the actual changes to field values that occurred. Methods of the HistoryWalker API retrieve the history data from the generated History Lines, instead of querying the `sys_audit` table.
- Using Audit table: In this case, the HistoryWalker API extracts data directly querying the `sys_audit` table.

By default, it populates the data to support the “changes”, “changesFrom” and “changesTo” methods in the walked record, as well as provides record and field level security. Additionally, it can enable journal fields and variables to be also populated in the walked GlideRecord when walking through the updates.

This API enables you to:

- apply the appropriate history/audit data to get an existing GlideRecord to the state it was in a specific update count.
- instruct the `HistoryWalker` API to use `sys_audit` table instead of `sys_history_set/sys_history_line` tables to retrieve its data.
- turn off row-level access control.
- turn off field-level access control.
- turn off retrieval and processing of “changes” data.
- Enable journal fields.
• Enable variables.

**HistoryWalker - HistoryWalker(String tableName, String sysId)**

Fetches the database record based on the parameters, using the History Sets to retrieve the historic data.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>tableName</td>
</tr>
<tr>
<td>sysId</td>
</tr>
</tbody>
</table>

Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
incGr.getUniqueValue());
if (hw.walkTo(3)) {
    var oldPriority = hw.getWalkedRecord().priority;
    gs.info('Incident priority in update number ' +
    hw.getUpdateNumber() + ' was ' + oldPriority);
} else
    gs.info('Incident does not have update number 3');
```

Output:

```
Incident priority in update number 3 was 4
```

**HistoryWalker - HistoryWalker(String tableName, String sysId, Boolean useAudit)**

Fetches the database record based on the parameters, using the History Sets or Audit data to retrieve the historic data, depending on the third parameter.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>tableName</td>
</tr>
<tr>
<td>sysId</td>
</tr>
</tbody>
</table>
| useAudit  | Boolean | • If set to true, uses audit data to retrieve historic date.  
• If set to false, uses history set to retrieve historic date. |
Example

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
    incGr.getUniqueValue(), true);
if (hw.walkTo(3)) {
    var oldPriority = hw.getWalkedRecord().priority;
    gs.info('Incident priority in update number ' +
        hw.getUpdateNumber() + ' was ' + oldPriority);
} else
    gs.info('Incident does not have update number 3');
```

Output:

```
Incident priority in update number 3 was 4
```

**HistoryWalker - getUpdateNumber()**

Gets the update number of the current walked glide record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Current update number or, -1 if record is not found</td>
</tr>
</tbody>
</table>

Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
    incGr.getUniqueValue());
hw.walkTo(3);
gs.info('Update number: ' + hw.getUpdateNumber());
```

Output:

```
Update number: 3
```

**HistoryWalker - getWalkedRecord()**

Gets the record filled with the history/audit data after walking to an update number.
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The walked GlideRecord.</td>
</tr>
</tbody>
</table>

#### Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
                                   incGr.getUniqueValue());
hw.walkTo(0);
var walkedRecord = hw.getWalkedRecord();
gs.info('Priority in update number 0: ' +
        walkedRecord.priority);

hw.walkTo(1);
walkedRecord = hw.getWalkedRecord();
gs.info('Short description in update number 1: ' +
        walkedRecord.short_description);
```

**Output:**

```
Priority in update number 0: 4
Short description in update number 1: My monitor has stopped working
```

### HistoryWalker - getWalkedRecordCopy()

Gets a copy of the record filled with the history/audit data after walking to an update number.

**Note:** The `getWalkedRecord()` API might modify the obtained walkedRecord after walking to another update number. The `getWalkedRecordCopy()` API gets a clone to the walked record to prevent that.
Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var walkedRecord = [];
var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
  incGr.getUniqueValue());
hw.walkTo(0);
walkedRecord[0] = hw.getWalkedRecordCopy();
hw.walkTo(1);
walkedRecord[1] = hw.getWalkedRecordCopy();

gs.info('Priority in update number 0: ' +
  walkedRecord[0].priority);
gs.info('Short description in update number 1: ' +
  walkedRecord[1].short_description);
```

Output:

```
Priority in update number 0: 4
Short description in update number 1: My monitor has stopped working
```

**HistoryWalker - isFieldLevelSecurity()**

Specifies if the record-level read access is applied on the record when retrieving from the database.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if field level security is enabled, else returns false.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
  incGr.getUniqueValue());
gs.info('Field level security is active: ' +
  hw.isFieldLevelSecurity());
```
Field level security is active: true

HistoryWalker - isRecordLevelSecurity()

Specifies if the record-level read access is applied on the record when retrieving from the database.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the record-level security is enabled, else returns false.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
  incGr.getUniqueValue());
gs.info('Record level security is active: ' +
  hw.isRecordLevelSecurity());
```

Output:

Record level security is active: true

HistoryWalker - isWithChanges()

Specifies if any of the methods that walk the record from one update to another, support the “changes” data for each element.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the changes support is enabled, else returns false.</td>
</tr>
</tbody>
</table>

### Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
                               incGr.getUniqueValue());
gs.info('Changes is active: ' + hw.isWithChanges());
```

Output:

Changes is active: true

---

**HistoryWalker - isWithJournalFields()**

Specifies if journal type fields are populated from the historical values.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if journal fields are populated, else returns false.</td>
</tr>
</tbody>
</table>

### Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
                               incGr.getUniqueValue());
gs.info('Populating journal fields is active: ' + hw.isWithJournalFields());
```

Output:

Populating journal fields is active: false
HistoryWalker - isWithVariables()

Specifies if values are set for variables that are recorded in the history.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if including values for variables, else returns false.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
                             incGr.getUniqueValue());
gs.info('Populating variables is active: ' + hw.
                             isWithVariables());
```

Output:

```
Populating variables is active: false
```

HistoryWalker - setFieldLevelSecurity(Boolean fieldLevelSecurity)

Sets the field-level read access on each element before setting the historical value of that element in the GlideRecord. If the field-level security is enabled, it prevents the API to populate the fields of the walked record that the user of the API does not have access to.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldLevelSecurity</td>
<td>Boolean</td>
<td>If set to true, field-level security is enabled. The default value is true.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
   incGr.getUniqueValue());
hw.setFieldLevelSecurity(false);
hw.walkTo(0);
```

**HistoryWalker - setRecordLevelSecurity(Boolean recordLevelSecurity)**

Sets the record-level read access on the record when retrieving from the database. The record-level security prevents the API to retrieve the walked record if the user of the API does not have access to the GlideRecord.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordLevelSecurity</td>
<td>Boolean</td>
<td>If set to true, record-level read access security is enabled. The default value is true.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
   incGr.getUniqueValue());
hw.setRecordLevelSecurity(false);
hw.walkTo(0);
```

**HistoryWalker - setWithChanges(Boolean withChanges)**

Sets the "changes" data support for each element for a method that walks the record from one update to another.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>withChanges</td>
<td>Boolean</td>
<td>If set to true, the &quot;changes&quot; data is supported for each element. The default value is true.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example: With the Changes data support

```javascript
var hw = new sn_hw.HistoryWalker(incGr.getTableName(),   
  incGr.getUniqueValue());
hw.walkTo(0);
do {
    printChangedFields(hw);
} while (hw.walkForward());

function printChangedFields(hw) {
    var walkedGr = hw.getWalkedRecord();
    var fields =
        GlideScriptRecordUtil.get(walkedGr).getChangedFieldNames();
    gs.print("Fields changed at update " + hw.getUpdateNumber() + 
        " were: ");
    for (var j = 0; j < fields.size(); j++)
        gs.print(" " + fields.get(j));
    gs.print(" ");
}
```

Example: Without the Changes data support

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');
var hw = new sn_hw.HistoryWalker(incGr.getTableName(),   
  incGr.getUniqueValue());
hw.setWithChanges(false);
hw.walkTo(0);
do {
    var oldPriority = hw.getWalkedRecord().priority;
    gs.print('Incident priority in update number ' + 
        hw.getUpdateNumber() + ' was ' + oldPriority);
} while (hw.walkForward());
```

**HistoryWalker - setWithJournalFields(Boolean withJournalFields)**

Specifies if journal type fields are populated from the historical values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>withJournalFields</td>
<td>Boolean</td>
<td>If set to true, include journal-type fields. The default value is false.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
    incGr.getUniqueValue());
hw.setWithJournalFields(true);
if (hw.walkTo(0)) {
    var workNotes = hw.getWalkedRecord().work_notes;
    gs.info('Work Notes in update number ' +
        hw.getUpdateNumber() + ' was ' + workNotes);
}
```

**HistoryWalker** - setWithVariables(Boolean withVariables)

Specifies if variables are populated from the historical values.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>withVariables</td>
<td>Boolean</td>
<td>If set to true, values are populated for variables. The default value is false.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
    incGr.getUniqueValue());
hw.setWithVariables(true);
hw.walkTo(0);
if (hw.walkTo(0)) {
    var varUrgency = hw.getWalkedRecord().variables.urgency;
    gs.info('Variable Urgency in update number ' +
        hw.getUpdateNumber() + ' was ' + varUrgency);
}
```
HistoryWalker - walkBackward()

Applies the appropriate history/audit data to get a walked GlideRecord to the state when it was one update number backward. If the previous update count is missing from the history/audit data, it will walk to the previous available update count.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if walking to the specified update number was possible. Else, returns false, for example if already walked to the update number 0.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
  incGr.getUniqueValue());
hw.walkTo(incGr.sys_mod_count);
do {
  var oldPriority = hw.getWalkedRecord().priority;
gs.info('Incident priority in update number ' +
  hw.getUpdateNumber() + ' was ' + oldPriority);
} while (hw.walkBackward())
```

Output:

```
Incident priority in update number 5 was 2
Incident priority in update number 4 was 4
Incident priority in update number 3 was 4
Incident priority in update number 2 was 4
Incident priority in update number 1 was 4
Incident priority in update number 0 was 4
```

HistoryWalker - walkForward()

Applies the appropriate history/audit data to get a walked GlideRecord to the state when it was one update number forward. If next update count is missing from the history/audit data, it will walk to the next available update count.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if walking to the specified update number was possible. Else, returns false, for example if already walked to the GlideRecord update count.</td>
</tr>
</tbody>
</table>

Example:

```js
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
    incGr.getUniqueValue());
hw.walkTo(0);
do {
    var oldPriority = hw.getWalkedRecord().priority;
    gs.info('Incident priority in update number ' +
        hw.getUpdateNumber() + ' was ' + oldPriority);
} while (hw.walkForward())
```

Output:

```
Incident priority in update number 0 was 4
Incident priority in update number 1 was 4
Incident priority in update number 2 was 4
Incident priority in update number 3 was 4
Incident priority in update number 4 was 4
Incident priority in update number 5 was 2
```

**HistoryWalker - walkTo(int updateCount)**

Applies the appropriate history/audit data to get a GlideRecord to the state it was in a specific update count. Use getWalkedRecord() or getWalkedRecordCopy() after walking to an update number to retrieve the "walked" GlideRecord.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>updateCount</td>
<td>Integer</td>
<td>The update number to walk to.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>true if walking to the specified update number was possible, false otherwise, for example if the requested update is greater than the update count of the GlideRecord, or if there is no history/audit data of the requested update number</td>
<td></td>
</tr>
</tbody>
</table>

Example:

```javascript
var incGr = new GlideRecord('incident');
incGr.get('number', 'INC0000015');

var hw = new sn_hw.HistoryWalker(incGr.getTableName(),
    incGr.getUniqueValue());
if (hw.walkTo(3)) {
    var oldPriority = hw.getWalkedRecord().priority;
    gs.info('Incident priority in update number ' +
        hw.getUpdateNumber() + ' was ' + oldPriority);
} else
    gs.info('Incident does not have update number 3');
```

Output:

Incident priority in update number 3 was 4

HostnameJS

Formats host names according to property settings.

Use with any server-side script when you need to format host names.

HostnameJS - format(String hostname, String source)

Formats the specified host name according to the property settings.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostname</td>
<td>String</td>
<td>The host name to format</td>
</tr>
<tr>
<td>source</td>
<td>String</td>
<td>The property settings source</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The system name</td>
</tr>
</tbody>
</table>

HostnameJS - getDomainName()

Returns the DNS domain name.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The domain name</td>
</tr>
</tbody>
</table>

```javascript
var hjs = new HostnameJS();
hjs.getDomainName();
```

HostnameJS - getSysName()

Returns the current system name.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The system name</td>
</tr>
</tbody>
</table>

```javascript
var hjs = new HostnameJS();
hjs.getSysName();
```

i18NV3

Provides methods to get and format translated messages.

The i18N methods are accessed using the g_i18n global object.

i18NV3 - getMessage(String msgKey, Function callback)

Retrieves a translated message.

If the specified string exists in the database for the current language, then the translated message is returned. If the specified string does not exist for the current language, then the English version of the string is returned. If the string does not exist at all in the database, then the ID itself is returned.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msgKey</td>
<td>String</td>
<td>The message to be retrieved.</td>
</tr>
<tr>
<td>callback</td>
<td>Function</td>
<td>The function to be called when the message has been retrieved. The callback function has one argument, a string that is the translated message.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**i18NV3 - getMessages(Array msgKeys, Function callback)**

Retrieves a set of messages.

If the specified string exists in the database for the current language, then the translated message is returned. If the specified string does not exist for the current language, then the English version of the string is returned. If the string does not exist at all in the database, then the ID itself is returned.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msgKeys</td>
<td>Array</td>
<td>An array of keys specifying the messages to be retrieved.</td>
</tr>
<tr>
<td>callback</td>
<td>Function</td>
<td>The function to be called when the messages have been retrieved. The callback function has one argument, an object containing key-value pairs, where key is the requested message key, and the value is the translated string.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**i18NV3 - format(String message, Object map)**

Formats a string containing named tokens with values from a map.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to have the tokens added.</td>
</tr>
<tr>
<td>map</td>
<td>Object</td>
<td>The map of name/value pairs to replace in the message.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The formatted string</td>
</tr>
</tbody>
</table>

```java
// Returns: "The rich young ruler was very very rich"
nowapi.g_i18n.format("The {p1} {p2} {p3} was very very {p1}",
    {p1: "rich", p2: "young", p3: "ruler"});
```

### ICalUtilSNC

The ICalUtilSNC API provides functions to generate iCalendar compliant events.

### ICalUtilSNC - formatICalComponent(Array arrEvents)

Generates a formatted VCALENDAR component.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrEvents</td>
<td>Array</td>
<td>Contains the properties that make up a VCALENDAR component.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>Contains properties needed to form a VCALENDAR.</td>
</tr>
</tbody>
</table>

### ICalUtilSNC - formatICalEvent(Array arrEvent, Boolean useAlarm)

Generates a formatted VEVENT.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrEvent</td>
<td>Array</td>
<td>Contains the individual properties that make up a VEVENT.</td>
</tr>
<tr>
<td>useAlarm</td>
<td>Boolean</td>
<td>If true, adds a VALARM to the VEVENT.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The formatted VEVENT.</td>
</tr>
</tbody>
</table>

**ICalUtilSNC - formatRecurringRule (Object ruleObj)**

Formats the RRULE property for a VEVENT.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruleObj</td>
<td>Object</td>
<td>Contains the properties for the RRULE property.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The formatted RRULE property.</td>
</tr>
</tbody>
</table>

**ICalUtilSNC - getDateFromScheduleDateTime (GlideScheduleDateTime scheduleDateTime)**

Returns the date from the GlideScheduleDateTime.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleDateTime</td>
<td>GlideScheduleDateTime</td>
<td>A GlideScheduleDateTime representation of the date and time.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Formatted date.</td>
</tr>
</tbody>
</table>
ICalUtilSNC - getSDT(String sdtStr, String timeZone)

If provided with a formatted date time string, it returns a GlideScheduleDateTime object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdtStr</td>
<td>String</td>
<td>Formatted ScheduleDateTime.</td>
</tr>
<tr>
<td>timeZone</td>
<td>String</td>
<td>If provided, will be used for the generated GlideScheduleDateTime.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideScheduleDateTime</td>
<td>A GlideScheduleDateTime representation of the sdtStr.</td>
</tr>
</tbody>
</table>

ICalUtilSNC - getTimeFromScheduleDateTime (GlideScheduleDateTime scheduleDateTime)

Returns the time from the GlideScheduleDateTime.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleDateTime</td>
<td>GlideScheduleDateTime</td>
<td>A GlideScheduleDateTime representation of the date and time.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Formatted time.</td>
</tr>
</tbody>
</table>

ICalUtil

The ICalUtil API is an extension of the ICalUtilSNC API.

IdentificationEngine

The IdentificationEngine uses the Identification and Reconciliation framework to minimize creation of duplicate CIs and to reconcile CI attributes by only accepting information from authorized data sources when updating the CMDB.

You must enable the Configuration Management for Scoped Apps (CMDB) (com.snc.cmdb.scoped) plugin in order to use this class.

When using this class in a scoped application, use the sn_cmdb namespace identifier.
IdentificationEngine - createOrUpdateCI(String source, String input)

Insert or update configuration items in the CMDB based on identification and reconciliation rules. Use this API instead of updating the CMDB directly.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>String</td>
<td>Identifies the data source of the CI information. These must be one of the choice values defined for the discovery_source field of the cmdb_ci table.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| input | String | A JSON formatted string of configuration items to be added or updated. Each input string is in the format ‘items: 
{(items)}, relations:{(relations)}, related{(related)}’, where each item within the items, relations, and related lists contains name-value pairs. The possible name-value pairs within the items list are:  
- className - the sys_class_name of the CI to be created or updated.  
- values:{} - the field information for the CI as name-value pairs, where the name is the field name. When updating reference fields, the value must be the referenced sys_id.  
- lookup:{()} - a list of records with each item having name-value pairs like the items list.  
- related: {()} - a list of records with each item having name-value pairs like the items list.  

The possible name-value pairs within the relations list are:  
- parent - index of the parent item in the dependency relation  
- child - index of the child item in the dependency relation  
- type - the relationship type. This is one of the name field values from the cmdb_rel_type table. |
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| String  | A JSON formatted string that is a list of results for the configuration items in the input string. Each result string is in the format  
  "items: ([]), relations: ([]):", where each item within the items and relations lists contains name-value pairs.

The possible name-value pairs within the items list are:

- **className**: the sys_class_name for the CI that was updated or created.
- **operation**, which is one of INSERT, UPDATE, UPDATE_WITH_UPGRADE, UPDATE_WITH_DOWNGRADE, UPDATE_WITH_SWITCH, DELETE, NO_CHANGE
- **sysId**: the sys_id of the CI that was updated or created.
- **relatedSysIds**: a list of sys_id values of CIs used during lookup based identification.
- **maskedAttributes**: a list of attributes whose update by a non-authoritative data source gets skipped as defined by the Reconciliation Rules.
- **identifierEntrySysId**: sys_id of identifier entry used during matching.
- **errors**: a list of errors in the format of (error, message string), where error can be ABANDONED, INVALID_INPUT_DATA, IDENTIFICATION_RULE_MISSING, IDENTIFICATION_RULE_FOR_LOOKUP_MISSING, NO_LOOKUP_RULES_FOR_DEPENDENT_CI, NO_CLASS_NAME_FOR_INDEPENDENT_CI, MISSING_DEPENDENCY, MULTIPLE_DEPENDENCIES, MULTIPLE_DUPLICATE_RECORDS, RELATION_CHAIN_ENDS_AT_QUALIFIER, QUALIFICATION_LOOP, TYPE_CONFLICT_IN_QUALIFICATION, MULTI_MATCH, REQUIRED_ATTRIBUTE_EMPTY, RECLASSIFICATION_NOT_ALLOWED
- **duplicateIndices**: a list of indexes of items that are duplicates of the current item.
- **identificationAttempts**: a list of attempts in the format of (attributes, identifierName, attemptResult, searchOnTable) where
  - **attributes**: the attributes of identifier entry used during identification
  - **identifierName**: the CI identifier to which this identifier entry belongs
  - **attemptResult**: one of SKIPPED, NO_MATCH, MATCHED, MULTI_MATCH
  - **searchOnTable**: the table searched during the identification process.
Identify an independent CI.

```javascript
var payload = {
    items: [{
        className:'cmdb_ci_linux_server',
        values: {
            name:'stry0900844 CI 2',
            serial_number:'9876EFGH',
            mac_address:'4653XYZ',
            ip_address:'10.10.10.4',
            ram:'1238'
        }
    }]
};

var jsonUntil = new JSON();
var input = jsonUntil.encode(payload);
var output = sn_cmdb.IdentificationEngine.createOrUpdateCI('ServiceNow', input);
gs.print(output);
```

Output:

```javascript
{"items": [{
    className:"cmdb_ci_linux_server", "operation":"INSERT",
    "sysId":"33dd7e760f750200293f8bece1050ee0", "identifierEntrySysId":"Unknown",
    "identificationAttempts": [{
        "attributes": {
            "serial_number"}, "identifierName": "Hardware Rule",
            "attemptResult": "NO_MATCH", "searchOnTable": "cmdb_ci_hardware"},
            {"attributes": [{"name"},
                "identifierName": "Hardware Rule", "attemptResult": "NO_MATCH", "searchOnTable": "cmdb_ci_hardware"}]
        },
        "relations": []
    }
]}
```

Identify a dependent CI.

```javascript
var payload = {
    items: [
        {className:'cmdb_ci_web_server',
            values: {
                name:'apache linux den 200',
                running_process_command: 'xyz',
                running_process_key_parameters: 'abc',
                tcp_port:'3452'
            }
        },
        {className:'cmdb_ci_linux_server',
            values: {
                name:'lnux100', ram:'2048'
            }
        }
    ],
    relations: [{parent: 0, child: 1, type: 'Runs on::Runs'}]
};

var jsonUntil = new JSON();
var input = jsonUntil.encode(payload);
var output = sn_cmdb.IdentificationEngine.createOrUpdateCI('ServiceWatch', input);
gs.print(output);
```

Output: line breaks have been added for presentation.

```javascript
{"items": [{
    className:"cmdb_ci_web_server", "operation":"UPDATE",
    "sysId":"5f8af237c0a8010e01a932999468b83a", "identifierEntrySysId":"8985a23ec3f0020000",
    "relations": [{parent: 0, child: 1, type: 'Runs on::Runs'}]
}]
```
Identify an independent CI with lookup based identification.

```javascript
var payload = {items: [
    {className:'cmdb_ci_netgear',
     values: {name:'ny8500-nbxs08',
              ports:'1200'},
     lookup: [{className:'cmdb_serial_number',
               values:{serial_number:'1234ABCD'},
               serial_number_type:'uuid',absent:'false',valid:'true'},
            {className:'cmdb_serial_number',
             values:{serial_number:'3456EFGH'},
             serial_number_type:'system',absent:'false',valid:'true'}]},

    var jsonUntil = new JSON();
    var input = jsonUntil.encode(payload);
    var output = 
      sn_cmdb.IdentificationEngine.createOrUpdateCI('ServiceNow',
                                                      input);
    gs.print(output);

Output: line breaks have been added for presentation.

{"items":
["className":"cmdb_ci_netgear","operation":"UPDATE","sysId":55b35562c0a8010e01cfa98f,"relatedSysIds":
  ["c9d5d426538202007c949096a11c0829","0dd5d426538202007c949096a11c0829"],
  "identifierEntrySysId":"c12f9be8c3400200d8d4bea192d3ae92",  
  "identificationAttempts":[]",
  "identifierName":"Hardware Rule",  
  "serial_number","identifierName":"Hardware Rule",  
  "identifierEntrySysId":556eb250c3400200d8d4bea192d3ae92,
  "identificationAttempts":[]",
  "attemptResult":"MATCHED","
  "identifierName":"Hardware Rule",
  "searchOnTable":"cmdb_ci_hardware"],
  "relations":[]
}]
```

**IdentificationEngine - identifyCI(String jsonString)**

Determines the operation (insert/update) that will be performed with the specified payload without committing the operation in the database.

This works just like `createOrUpdateCI()`, but does not commit the result.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| jsonString | String | A JSON formatted string of configuration items to be added or updated. Each input string is in the format 'items: ({}), relations:({}), where each item within the items and relations lists contains name-value pairs.

The possible name-value pairs within the items list are:
- className - the sys_class_name of the CI to be created or updated.
- values:() - the field information for the CI as name-value pairs, where the name is the field name.
- lookup:() - a list of records with each item having name-value pairs like the items list.

The possible name-value pairs within the relations list are:
- parent - index of the parent item in the dependency relation
- child - index of the child item in the dependency relation
- type - the relationship type. This is one of the name field values from the cmdb_rel_type table. |
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| String | A JSON formatted string that is a list of results. Each result string is in the format of \`items: \{\}, relations:\{\}\`, where each item within the items and relations lists contains name-value pairs. The possible name-value pairs within the items list are:  
  - **className** - the sys_class_name for the CI that was updated or created.  
  - **operation**, which is one of INSERT, UPDATE, UPDATE_WITH_UPGRADE, UPDATE_WITH_DOWNGRADE, UPDATE_WITH_SWITCH, DELETE, NO_CHANGE.  
  - **sysId** - the sys_id of the CI that was updated or created.  
  - **relatedSysIds** - a list of sys_id values of CIs used during lookup based identification.  
  - **identifierEntrySysId** - sys_id of identifier entry used during matching.  
  - **errors** - a list of errors in the format of \((error, message string)\), where error can be ABANDONED, INVALID_INPUT_DATA, IDENTIFICATION_RULE_MISSING, IDENTIFICATION_RULE_FOR_LOOKUP_MISSING, NO_LOOKUP_RULES_FOR_DEPENDENT_CI, NO_CLASS_NAME_FOR_INDEPENDENT_CI, MISSING_DEPENDENCY, MULTIPLE_DEPENDENCIES, MULTIPLE_DUPLICATE_RECORDS, RELATION_CHAIN_ENDS_AT_QUALIFIER, QUALIFICATION_LOOP, TYPE_CONFLICT_IN_QUALIFICATION, MULTI_MATCH, REQUIRED_ATTRIBUTE_EMPTY, RECLASSIFICATION_NOT_ALLOWED.  
  - **duplicateIndices** - a list of indexes of items that are duplicates of the current item.  
  - **identificationAttempts** - a list of attempts in the format of \((attributes, identifierName, attemptResult, searchOnTable)\) where  
    - **attributes** - the attributes of identifier entry used during identification  
    - **identifierName** - the CI identifier to which this identifier entry belongs  
    - **attemptResult** - one of SKIPPED, NO_MATCH, MATCHED, MULTI_MATCH  
    - **searchOnTable** - the table searched during the identification process.  
  
  The possible name-value pairs within the relations list are:  
  - **className** - the relationship CI's class name  
  - **sysId** - the sys_id of the relationship CI inserted or updated.
IdentificationEngine - runIdentificationAudit(GlideRecord gr)

Run an identification audit against the specified CI to detect duplicates. If duplicates are found, duplication tasks are created. Only use this method on CI types with independent identification rules.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>The CI on which to run the audit to detect duplicates. The CI must have independent identification rules.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

IdentificationEngineScriptableApi

The IdentificationEngineScriptableApi uses the Identification and Reconciliation framework to minimize creation of duplicate CIs and to reconcile CI attributes by only accepting information from authorized data sources when updating the CMDB.

IdentificationEngineScriptableApi - createOrUpdateCI( String source, String input)

Insert or update configuration items in the CMDB based on identification and reconciliation rules. Use this API instead of updating the CMDB directly.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>String</td>
<td>Identifies the data source of the CI information. These must be one of the choice values defined for the discovery_source field of the cmdb_ci table.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| input | String | A JSON formatted string of configuration items to be added or updated. Each input string is in the format: `items: [{}], relations: [{}], related: [{}]`, where each item within the items, relations, and related lists contains name-value pairs. The possible name-value pairs within the items list are:  
  - className - the sys_class_name of the CI to be created or updated.  
  - values: {} - the field information for the CI as name-value pairs, where the name is the field name. When updating reference fields, the value must be the referenced sys_id.  
  - lookup: [{}], related: [{}], where each item has name-value pairs like the items list.  
  - parent - index of the parent item in the dependency relation  
  - child - index of the child item in the dependency relation  
  - type - the relationship type. This is one of the name field values from the cmdb_rel_type table. |
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| String| A JSON formatted string that is a list of results for the configuration items in the input string. Each result string is in the format '{items: [{}], relations: [{}]}', where each item within the items and relations lists contains name-value pairs. The possible name-value pairs within the items list are:  
  - className - the sys_class_name for the CI that was updated or created.  
  - operation, which is one of INSERT, UPDATE, UPDATE_WITH_UPGRADE, UPDATE_WITH_DOWNGRADE, UPDATE_WITH_SWITCH, DELETE, NO_CHANGE  
  - sysId - the sys_id of the CI that was updated or created.  
  - relatedSysIds - a list of sys_id values of CIs used during lookup based identification.  
  - maskedAttributes - a list of attributes whose update by a non-authoritative data source gets skipped as defined by the Reconciliation Rules.  
  - identifierEntrySysId - sys_id of identifier entry used during matching.  
  - errors - a list of errors in the format of (error, message string), where error can be ABANDONED, INVALID_INPUT_DATA, IDENTIFICATION_RULE_MISSING, IDENTIFICATION_RULE_FOR_LOOKUP_MISSING, NO_LOOKUP_RULES_FOR_DEPENDENT_CI, NO_CLASS_NAME_FOR_INDEPENDENT_CI, MISSING_DEPENDENCY, MULTIPLE_DEPENDENCIES, MULTIPLE_DUPLICATE_RECORDS, RELATION_CHAIN_ENDS_AT_QUALIFIER, QUALIFICATION_LOOP, TYPE_CONFLICT_IN_QUALIFICATION, MULTI_MATCH, REQUIRED_ATTRIBUTE_EMPTY, RECLASSIFICATION_NOT_ALLOWED  
  - duplicateIndices - a list of indexes of items that are duplicates of the current item.  
  - identificationAttempts - a list of attempts in the format of (attributes, identifierName, attemptResult, searchOnTable) where  
    - attributes - the attributes of identifier entry used during identification  
    - identifierName - the CI identifier to which this identifier entry belongs  
    - attemptResult - one of SKIPPED, NO_MATCH, MATCHED, MULTI_MATCH  
    - searchOnTable - the table searched during the identification process.  |
ServiceNow

Kingston

Now Platform Custom Business Applications

Identify an independent CI.
var payload = {items: [{className:'cmdb_ci_linux_server',
values: {name:'stry0900844 CI 2',
serial_number:'9876EFGH',
mac_address:'4653XYZ',
ip_address:'10.10.10.4',
ram:'1238'}}]
};
var jsonUntil = new JSON();
var input = jsonUntil.encode(payload);
var output =
SNC.IdentificationEngineScriptableApi.createOrUpdateCI('ServiceNow',
input);
gs.print(output);
Output:
{"items":
[{"className":"cmdb_ci_linux_server","operation":"INSERT",
"sysId":"33dd7e760f750200293f8fbce1050ee0","identifierEntrySysId":"Unknown",
"identificationAttempts":[{"attributes":
["serial_number"],"identifierName":"Hardware Rule",
"attemptResult":"NO_MATCH","searchOnTable":"cmdb_ci_hardware"},
{"attributes":["name"],
"identifierName":"Hardware
Rule","attemptResult":"NO_MATCH","searchOnTable":"cmdb_ci_hardware"}]}
],"relations":[]}

Identify a dependent CI.
var payload =
{items: [
{className:'cmdb_ci_web_server',
values: {name:'apache linux den 200',
running_process_command: 'xyz',
running_process_key_parameters:
'abc',
tcp_port:'3452'}},
{className:'cmdb_ci_linux_server',
values: {name:'lnux100', ram:'2048'}}],
relations:[{parent: 0, child: 1, type: 'Runs
on::Runs'}]
};
var jsonUntil = new JSON();
var input = jsonUntil.encode(payload);
var output =
SNC.IdentificationEngineScriptableApi.createOrUpdateCI('ServiceWatch',
input);
gs.print(output);
Output: line breaks have been added for presentation.
{"items":[
{"className":"cmdb_ci_web_server","operation":"UPDATE",

"sysId":"5f8af237c0a8010e01a932999468b83a","identifierEntrySysId":"8985a23ec3f002
© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States
and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.

1220


ServiceNow

Kingston

Now Platform Custom Business Applications

"identificationAttempts":[{"attributes":
["running_process_command","running_process_key_parameters"],
"identifierName":"Application
Rule","attemptResult":"MATCHED","searchOnTable":"cmdb_ci_appl"}]},
{"className":"cmdb_ci_linux_server","operation":"UPDATE","sysId":"53958ff0c0a80164
"identifierEntrySysId":"556eb250c3400200d8d4bea192d3ae92",
"identificationAttempts":[{"attributes":
["serial_number"],"identifierName":"Hardware Rule",
"attemptResult":"SKIPPED","searchOnTable":"cmdb_ci_hardware"},
{"attributes":["name"],
"identifierName":"Hardware
Rule","attemptResult":"MATCHED","searchOnTable":"cmdb_ci_hardware"}]}],
"relations":[{"className":"cmdb_rel_ci","operation":"UPDATE",
"sysId":"60bcdcebc0a8010e0124fb16964b989f"}]}

Identify an independent CI with lookup based identification.
var payload = {items: [
{className:'cmdb_ci_netgear',
values: {name:'ny8500-nbxs08',
ports:'1200'},
lookup: [{className:'cmdb_serial_number',
values:{serial_number:'1234ABCD',
serial_number_type:'uuid',absent:'false',valid:'true'}},
{className:'cmdb_serial_number',
values:{serial_number:'3456EFGH',
serial_number_type:'system',absent:'false',valid:'true'}}]}]};
var jsonUntil = new JSON();
var input = jsonUntil.encode(payload);
var output =
SNC.IdentificationEngineScriptableApi.createOrUpdateCI('ServiceNow',
input);
gs.print(output);
Output: line breaks have been added for presentation.

{"items":
[{"className":"cmdb_ci_netgear","operation":"UPDATE","sysId":"55b35562c0a8010e01cf
"relatedSysIds":
["c9d5d426538202007c949096a11c0829","0dd5d426538202007c949096a11c0829"],
"identifierEntrySysId":"c12f9be8c3400200d8d4bea192d3aea6",
"identificationAttempts":
[{"attributes":["serial_number","serial_number_type"],
"identifierName":"Hardware Rule","attemptResult":"MATCHED",
"searchOnTable":"cmdb_serial_number"}]}],
"relations":[]}

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States
and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.

1221


 Scoped equivalent

To use the `createOrUpdateCI(String source, String input)` method in a scoped application, use the corresponding scoped `IdentificationEngine` method:
`createOrUpdateCI(String source, String input)`.

**IdentificationEngineScriptableApi - identifyCI(String jsonString)**

Determines the operation (insert/update) that will be performed with the specified payload without committing the operation in the database.

This works just like `createOrUpdateCI()`, but does not commit the result.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| jsonString | String| A JSON formatted string of configuration items to be added or updated. Each input string is in the format 'items: [{}], relations:[{}]', where each item within the items and relations lists contains name-value pairs.

The possible name-value pairs within the items list are:

- className - the `sys_class_name` of the CI to be created or updated.
- values:{} - the field information for the CI as name-value pairs, where the name is the field name.
- lookup:[{}] - a list of records with each item having name-value pairs like the items list.

The possible name-value pairs within the relations list are:

- parent - index of the parent item in the dependency relation
- child - index of the child item in the dependency relation
- type - the relationship type. This is one of the name field values from the `cmdb_rel_type` table.
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| String | A JSON formatted string that is a list of results. Each result string is in the format `{items: [{}], relations: [{}]}, where each item within the items and relations lists contains name-value pairs. The possible name-value pairs within the items list are:
  - className - the sys_class_name for the CI that was updated or created.
  - operation, which is one of INSERT, UPDATE, UPDATE_WITH_UPGRADE, UPDATE_WITH_DOWNGRADE, UPDATE_WITH_SWITCH, DELETE, NO_CHANGE
  - sysId - the sys_id of the CI that was updated or created.
  - relatedSysIds - a list of sys_id values of CIs used during lookup based identification.
  - identifierEntrySysId - sys_id of identifier entry used during matching.
  - errors - a list of errors in the format of (error, message string), where error can be ABANDONED, INVALID_INPUT_DATA, IDENTIFICATION_RULE_MISSING, IDENTIFICATION_RULE_FOR_LOOKUP_MISSING, NO_LOOKUP_RULES_FOR_DEPENDENT_CI, NO_CLASS_NAME_FOR_INDEPENDENT_CI, MISSING_DEPENDENCY, MULTIPLE_DEPENDENCIES, MULTIPLE_DUPLICATE_RECORDS, RELATION_CHAIN_ENDS_AT_QUALIFIER, QUALIFICATION_LOOP, TYPE_CONFLICT_IN_QUALIFICATION, MULTI_MATCH, REQUIRED_ATTRIBUTE_EMPTY, RECLASSIFICATION_NOT_ALLOWED
  - duplicateIndices - a list of indexes of items that are duplicates of the current item.
  - identificationAttempts - a list of attempts in the format of (attributes, identifierName, attemptResult, searchOnTable) where
    - attributes - the attributes of identifier entry used during identification
    - identifierName - the CI identifier to which this identifier entry belongs
    - attemptResult - one of SKIPPED, NO_MATCH, MATCHED, MULTI_MATCH
    - searchOnTable - the table searched during the identification process.

The possible name-value pairs within the relations list are:
  - className - the relationship CI's class name
Scoped equivalent

To use the `identifyCI(String jsonString)` method in a scoped application, use the corresponding scoped IdentificationEngine method: `identifyCI(String jsonString)`.

IdentificationEngineScriptableApi - `runIdentificationAudit(GlideRecord gr)`

Run an identification audit against the specified CI to detect duplicates.
If duplicates are found, duplication tasks are created. Only use this method on CI types with independent identification rules.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>The CI on which to run the audit to detect duplicates. The CI must have independent identification rules.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Scoped equivalent

To use the `runIdentificationAudit(GlideRecord gr)` method in a scoped application, use the corresponding scoped IdentificationEngine method: `runIdentificationAudit(GlideRecord gr)`.

IDResult

A container class for the result of a CI Identifier.
Use with any server-side discovery script.

IDResult - `IDResult(Boolean explore, String sysID)`

Creates an instance of the IDResult class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explore</td>
<td>Boolean</td>
<td>Set to true if the CI should be explored; otherwise, false.</td>
</tr>
<tr>
<td>sysID</td>
<td>String</td>
<td>An existing CI's sys_id, or null if no existing CI could be found.</td>
</tr>
</tbody>
</table>
IPAddressFixup

After a device has been successfully discovered, ensures that no other device has the same IP address. If any duplicates are found, the IP address field is cleared.

Use with any server-side Discovery script to validate IP addresses.

**IPAddressFixup - dedupe(String tableName, String ip)**

Removes duplicates of the specified IP address in the specified table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table to check for duplicates</td>
</tr>
<tr>
<td>ip</td>
<td>String</td>
<td>The IP address to check for</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**IPAddressFixup - dedupeAll()**

Removes all duplicate IP addresses from the tables.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**IPAddressFixup - fix()**

Removes all duplicate IP addresses and ensures that the parent ip_address record is set to one of the NIC’s IP addresses.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**IPAddressFixup - getParentIP()**

Returns the parent IP address for the current device.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The parent IP address</td>
</tr>
</tbody>
</table>

**IPAddressFixup - syncIP()**

Ensures that the parent ip_address record is set to one of the NIC’s IP addresses, or leaves it alone if there were no NICs.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**IPAddressFixup - setParentIP(String ip)**

Sets the IP address field for the current CI.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>String</td>
<td>The IP address for the current CI.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
**IPService**

Encapsulates an IP Service.

Use this class during the discovery scanning phase.

**IPService - creates**

The table where this service creates entries.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>creates</td>
<td>String</td>
<td>The table where this service creates entries.</td>
</tr>
</tbody>
</table>

**IPService - description**

Description of the IPService.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>description</td>
<td>String</td>
<td>Description of the IPService.</td>
</tr>
</tbody>
</table>

**IPService - getFromArrayList(Array list)**

Returns an array of IPService instances specified by a Java ArrayList of sys_ids.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>Array</td>
<td>List of sys_ids.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>IPService instances</td>
</tr>
</tbody>
</table>

**IPService - IPService(Object source)**

Creates an instance of the IPService class.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Object</td>
<td>Either a GlideRecord instance or a sys_id string</td>
</tr>
</tbody>
</table>
**IPService - port**
The TCP or UDP port used by the service.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>String</td>
<td>The TCP or UDP port used by the service.</td>
</tr>
</tbody>
</table>

**IPService - protocol**
The protocol used by the service ("UDP", "TCP", or "TCP/UDP").

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>String</td>
<td>The protocol used by the service (&quot;UDP&quot;, &quot;TCP&quot;, or &quot;TCP/UDP&quot;).</td>
</tr>
</tbody>
</table>

**IPService - name**
A short name or handle for the IPService.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the IPService</td>
</tr>
</tbody>
</table>

**IPService - serviceName**
A long, descriptive English name for the IPService.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>String</td>
<td>A long, descriptive English name for the IPService.</td>
</tr>
</tbody>
</table>

**IPService - sysID**
The sys_id of this record.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysID</td>
<td>String</td>
<td>The sys_id of this record.</td>
</tr>
</tbody>
</table>
JavascriptProbe
Provides methods for using the JavascriptProbe on the MID server.
Use these methods in server scripts to send a probe to the MID server from JavaScript.

JavascriptProbe - addParameter(String name, String value)
Adds a parameter with the specified value to the probe.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The parameter name</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The parameter's value</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var jspr = new JavascriptProbe('msserv');
jspr.setName('jsprobe100');
jspr.setSource('jspr_source');
jspr.addParameter('support_feat', 'yes');
```

JavascriptProbe - create()
Creates the probe for the current MID server, puts it on the output queue, and sets its state to "ready".

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var jspr = new JavascriptProbe('msserv');
jspr.setName('jsprobe100');
jspr.setSource('jspr_source');
jspr.setJavascript('jspr_script');
```
**JavascriptProbe - JavascriptProbe(String midServer)**

Creates an instance of the JavascriptProbe.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>midServer</td>
<td>String</td>
<td>The MID server name</td>
</tr>
</tbody>
</table>

**JavascriptProbe - setJavascript(String script)**

Adds a script to the probe.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>script</td>
<td>String</td>
<td>The script to add to the probe</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var jspr = new JavascriptProbe('msserv');
jspr.setName('jsprobe100');
jspr.setSource('jspr_source');
jspr.setJavascript('jspr_script');
```

**JavascriptProbe - setName(String name)**

Sets the name of the JavascriptProbe

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name to give the JavascriptProbe.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
var jspr = new JavascriptProbe('msserv');
jspr.setName('jsprobe100');

**JavascriptProbe - setSource(String ip)**

Sets the source of the probe.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>String</td>
<td>The IP address of the target probe.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var jspr = new JavascriptProbe('msserv');
jspr.setName('jsprobe100');
jspr.setSource('100.100.100.1');

**J2js**

2JS script include allows you to convert Java objects to JavaScript objects.

If the given value is a Java object that can be converted to an equivalent JavaScript object, that conversion is performed and the result is returned. Otherwise the original Java object is returned.

The `j2js` class is available to server-side scripts.

The specific conversion performed in the order they are checked.

- Java String -> JavaScript string
- Java Boolean -> JavaScript boolean
- Java Integer -> JavaScript number
- Java Long -> JavaScript number
- Java Double -> JavaScript number
- Java Byte -> JavaScript number
- Java Float -> JavaScript number
- Java Short -> JavaScript number
- Java Character -> JavaScript number
- Java array -> JavaScript Array with order preserved
- Java List -> JavaScript Array with order preserved
- Java Map -> JavaScript Object with the key/value pairs translated into property/value pairs
- Java Set -> JavaScript Array in arbitrary order
Conversions are performed recursively on the elements of arrays, lists, or collections. For example, given a Java ArrayList of ArrayLists of Strings, this will return a JavaScript Array of Arrays of strings.

**J2js - j2js(Object javaObject)**

Converts a java object from system code to a JavaScript object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>javaObject</td>
<td>Object</td>
<td>A Java object from system code such as a Packages call.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>A JavaScript object if the parameter can be converted, otherwise it returns the Java object.</td>
</tr>
</tbody>
</table>

```javascript
var tu = new TableUtils("cmdb_ci_win_server");
var classes = tu.getHierarchy();
//getHierarchy returns a Java ArrayList, which is not exactly like a JavaScript Array
//for example you cannot get length
gs.print("classes = " + classes);
gs.print("classes.length = " + classes.length);

//convert to a JavaScript Array
gs.include("j2js");
var jsClasses = j2js(classes);
gs.print("jsClasses = " + jsClasses);
gs.print("jsClasses.length = " + jsClasses.length);
```

Output:

```javascript
classes = [cmdb_ci_win_server, cmdb_ci_server, cmdb_ci_computer,
           cmdb_ci_hardware, cmdb_ci]
classes.length = undefined
jsClasses =
            cmdb_ci_win_server, cmdb_ci_server, cmdb_ci_computer, cmdb_ci_hardware, cmdb_ci
jsClasses.length = 5
```

**JSON**

Provides methods to create JSON objects from a string, and to turn JSON objects into strings.

The JSON API has dynamic and static methods. You access the dynamic methods by creating a JSON object. To use the dynamic methods in a scoped application, add the global prefix when calling the constructor. You access the static methods by using the static JSON object.

The JavaScript ES5 native JSON object is used instead of the JSON static methods. If your script needs the old behavior, use the `encode()` and `decode()` methods.
This example creates a JSON object.

```javascript
// obj is a JSON formatted object
var parser = new JSON(); // encode() and decode() are dynamic methods.
var str = parser.encode(obj);
```

The `encode()` and `decode()` methods are deprecated. Use the JavaScript JSON object instead. This example creates a JSON object in a scoped script.

```javascript
// obj is a JSON formatted object
var parser = new global.JSON();
var str = parser.encode(obj);
```

This example uses the static methods.

```javascript
// obj is a JSON formatted object
var str = JSON.stringify(obj);
// str is a string containing content to be turned into a JSON formatted object
var obj = JSON.parse(str);
```

### JSON - `decode(String str)`

Creates an object or primitive type from a JSON formatted string.

**Note:** This class is deprecated. Use the JavaScript JSON object instead.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>String</td>
<td>A JSON formatted string.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object created from the specified string.</td>
</tr>
</tbody>
</table>

```javascript
var str = '{"name":"George","lastname":"Washington"}';
var parser = new JSON();
var obj = parser.decode(str);
gs.info('The first name is ' + obj.name);
```

Output: The first name is George

To use in a scoped script.

```javascript
var str = '{"name":"George","lastname":"Washington"}';
var parser = new global.JSON();
var obj = parser.decode(str);
gs.info('The first name is ' + obj.name);
```
Output: The first name is George

JSON - encode(Object jsonObject)

Creates a string from a JSON object.

**Note:** This class is deprecated. Use the JavaScript JSON object instead.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jsonObject</td>
<td>Object</td>
<td>The JSON object to be turned into a string.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string.</td>
</tr>
</tbody>
</table>

```javascript
var obj = {"name":"George","lastname":"Washington"};
var parser = new JSON();
var str = parser.encode(obj);
gs.info('The object ' + str);
```

Output: The object {"lastname":"Washington","name":"George"}

To use in a scoped script.

```javascript
var obj = {"name":"George","lastname":"Washington"};
var parser = new global.JSON();
var str = parser.encode(obj);
gs.info('The object ' + str);
```

JSON - JSON()

Creates an instance of the JSON class.

This class is deprecated. Use the JavaScript JSON object instead.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

JSON - parse(String str)

Creates an object or primitive type from a JSON formatted string.
The JavaScript ES5 native JSON object is used instead of the JSON static methods. If your script needs the old behavior, use the `encode()` and `decode()` methods.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>String</td>
<td>A JSON formatted string.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object created from the specified string.</td>
</tr>
</tbody>
</table>

```javascript
var str = '{"name":"George","lastname":"Washington"}';
var obj = JSON.parse(str);
gs.info('The first name is ' + obj.name);
```

Output: The first name is George

**JSON - stringify(Object jsonObject)**

Creates a string from a JSON object.

The `JSON.stringify()` method can only convert numbers, strings, and Java native objects to strings. It cannot convert user-defined objects to strings, unless those objects provide a `toJSON()` method. The call to `current.sys_id()` returns a GlideElement object which does not have a `toJSON()` method, so the return value for `stringify` is empty: `{}`. The JavaScript ES5 native JSON object is used instead of the JSON static methods. If your script needs the old behavior, use the `encode()` and `decode()` methods.

`JSON.stringify()` converts a value to JSON notation using the following guidelines:

- If the value has a `toJSON()` method, it is responsible for defining the data that is serialized.
- Boolean, number, and string objects are converted to the corresponding primitive values during stringification; in accordance with the traditional conversion semantics.
- If a function, undefined, or a symbol is encountered during conversion, it is either omitted (when it is found in an object) or censored to null (when it is found in an array). `JSON.stringify()` also returns undefined when passing in “pure” values, such as `JSON.stringify(function(){})` or `JSON.stringify(undefined)`.
- All symbol-keyed properties are ignored, even when using a `replacer()` function.
- Instances of Date implement the `toJSON()` function by returning a string (the same as `date.toISOString()`), thus they are treated as strings.
- The numbers Infinity and NaN, as well as the value null, are all considered null.
- For all other object instances, only their enumerable properties are serialized.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jsonObject</td>
<td>Object</td>
<td>JSON object to be turned into a string.</td>
</tr>
</tbody>
</table>
**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>JSON formatted string.</td>
</tr>
</tbody>
</table>

```javascript
var obj = {"name":"George","lastname":"Washington"};
var str = JSON.stringify(obj);
gs.info('The object ' + str);
```

Output: The object `{"name":"George","lastname":"Washington"}`

It is also possible to define a `replacer()` function and use that in the `stringify()` call. This function leverages the `GlideElement.toString()` method to provide a string representation of the GlideElement object.

```javascript
function replacer(name, val) {
  // convert GlideElement to string
  if ( val && val.constructor === GlideElement ) {
    return val.toString();
  } else {
    return val; // return as is
  }
}

var s = global.JSON.stringify(json_obj, replacer);
gs.info("json="+s);
```

**JSONParser**

Provides a JSON parser that does not use the JavaScript `eval()` function.

The methods of JSONParser do not validate the JSON string.

This class cannot be used in a scoped application. Use the JavaScript JSON object instead.

> **Note:** This class is deprecated. Use the JavaScript JSON object instead.

**JSONParser - JSONParser()**

Creates a JSONParser object.

This class cannot be used in a scoped application. Use the JavaScript JSON object instead.
This class is deprecated. Use the JavaScript JSON object instead.
JSON - Scoped

Provides scoped methods to create JSON objects from a string, and to turn JSON objects into strings.

For scoped applications, the JSON API uses static methods that call the JavaScript ES5 native JSON object.

Scoped JSON - parse(String str)

Creates an object or primitive type from a JSON formatted string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>String</td>
<td>A JSON formatted string.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object created from the specified string.</td>
</tr>
</tbody>
</table>

```javascript
var str = '{"name":"George","lastname":"Washington"}';
var obj = JSON.parse(str);
gs.info('The first name is ' + obj.name);
```

Output:

The first name is George

Scoped JSON - stringify(Object jsonObject)

Creates a string from a JSON object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jsonObject</td>
<td>Object</td>
<td>The JSON object to be turned into a string.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string.</td>
</tr>
</tbody>
</table>

```javascript
var obj = {"name":"George","lastname":"Washington"};
var str = JSON.stringify(obj);
```
gs.info('The object' + str);

Output:

The object {"name":"George","lastname":"Washington"}

JSUtil

JSUtil is a class of shortcuts for common JavaScript routines.

Script includes and business rules that are marked as Application = 'global' and Accessible from = 'All applications’ can be used in scoped scripts.

JSUtil is not available in scoped scripts.

The JSUtil API is available in server-side scripts.

JSUtil - doesNotHave(Object item)

Checks if item is null or is undefined.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>Object</td>
<td>The object to check</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the specified object is null or undefined.</td>
</tr>
</tbody>
</table>

var x = "the quick brown fox";
var y = ""
var z;

gs.print("x = " + x + ", JSUtil.doesNotHave(x) = " + JSUtil.doesNotHave(x)));
gs.print("y = " + y + ", JSUtil.doesNotHave(y) = " + JSUtil.doesNotHave(y));
gs.print("z = " + z + ", JSUtil.doesNotHave(z) = " + JSUtil.doesNotHave(z))

Output:

x = 'the quick brown fox', JSUtil.doesNotHave(x) = false
y = '', JSUtil.doesNotHave(y) = false
z = 'undefined', JSUtil.doesNotHave(z) = true

JSUtil - escapeAttr(String text)

Escapes ampersands commonly used to define URL attributes.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>String</td>
<td>The text</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The text with ampersands properly escaped.</td>
</tr>
</tbody>
</table>

```javascript
var attr = "sysparm_query=active=true&sysparm_view=special";
gs.print(JSUtil.escapeAttr(attr));
```

Output: This is the returned text. If the text is displayed in the application, the page will render the escaped ampersand with a single ampersand.

```javascript
sysparm_query=active=true&sysparm_view=special
```

**JSUtil - escapeText(String text)**

Escapes invalid XML characters such as "< > &".

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>String</td>
<td>The text</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The text with escape characters added.</td>
</tr>
</tbody>
</table>

```javascript
var html = "<b>This is my title</b>";
gs.print(JSUtil.escapeText(html));
```

Output: This is the value returned. If the result is displayed in the application, the page renders the brackets back so it appears that it is not escaped.

```javascript
<b>This is my title</b>
```

**JSUtil - getBooleanValue(GlideRecord gr, String field)**

Returns the value in a boolean GlideRecord field.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>A GlideRecord</td>
</tr>
<tr>
<td>field</td>
<td>String</td>
<td>The field from which to retrieve the boolean value.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if value of field is true, &quot;true&quot;, 1, or &quot;1&quot;.</td>
</tr>
</tbody>
</table>

```javascript
var inc = new GlideRecord("incident");
// get an active incident
inc.addActiveQuery();
inc.setLimit(1);
inc.query();
inc.next();

gs.print(JSUtil.getBooleanValue(inc, "active"));
```

Output: true

### JSUtil - has(Object item)

Checks if item is not null and is not undefined.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>Object</td>
<td>The Object to check</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the specified object is not null and is not undefined.</td>
</tr>
</tbody>
</table>

```javascript
var x = "the quick brown fox";
var y = "";
var z;

gs.print("x = ", JSUtil.has(x) = " + JSUtil.has(x));
gs.print("y = ", JSUtil.has(y) = " + JSUtil.has(y));
gs.print("z = ", JSUtil.has(z) = " + JSUtil.has(z));
```
Output:

\[
\begin{align*}
x &= 'the quick brown fox', \text{JSUtil.has}(x) = true \\
y &= '', \text{JSUtil.has}(y) = true \\
z &= 'undefined', \text{JSUtil.has}(z) = false
\end{align*}
\]

**JSUtil - instance_of(Object item, String class)**

Checks to see if the specified object is a member of the specified class.

For JavaScript objects, this method behaves exactly like the JavaScript operator "instanceof", but also supports Java objects.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>Object</td>
<td>The object to check</td>
</tr>
<tr>
<td>class</td>
<td>String</td>
<td>The class to check</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the specified object is a member of the specified class.</td>
</tr>
</tbody>
</table>

```javascript
var a = ['a','b','c'];
var b = 10;
var c = new GlideRecord("incident");
gs.print("JSUtil.instance_of(a,'Array') = " + JSUtil.instance_of(a,Array));
gs.print("JSUtil.instance_of(a,'String') = " + JSUtil.instance_of(a,String));
gs.print("JSUtil.instance_of(b,'String') = " + JSUtil.instance_of(b,String));
gs.print("JSUtil.instance_of(c,'GlideRecord') = " + JSUtil.instance_of(c,GlideRecord));
```

Output:

JSUtil.instance_of(a,'Array') = true
JSUtil.instance_of(a,'String') = false
JSUtil.instance_of(b,'String') = false
JSUtil.instance_of(c,'GlideRecord') = true

**JSUtil - isJavaObject(Object value)**

Checks if the specified object is a Java class.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Object</td>
<td>The object to check</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the specified object is an instance of a Java class.</td>
</tr>
</tbody>
</table>

```javascript
var tu = new TableUtils("incident");
var classes = tu.getHierarchy(); //Java ArrayList
var tables = ["task, incident"]; //JavaScript Array

gs.print("JSUtil.isJavaObject(classes) = " +
JSUtil.isJavaObject(classes));
gs.print("JSUtil.isJavaObject(tables) = " +
JSUtil.isJavaObject(tables));
```

Output:

```
JSUtil.isJavaObject(classes) = true
JSUtil.isJavaObject(tables) = false
```

**JSUtil - logObject(Object obj, String name)**

Logs all the properties in the given object: name, type, and value.

Output is written to the console if you are running from a background script or have debug logging enables. The output is also written to the system log.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>Object</td>
<td>The object for which to enumerate properties</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Optional name for the logged object</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var arr = ["a","b","c"];`
var inc = new GlideRecord("incident");
//get an active incident
inc.addActiveQuery();
inc.setLimit(1);
inc.query();
inc.next();

JSUtil.logObject(arr, "arr");
JSUtil.logObject(inc, "inc");

Output:

Log Object: arr
Array of 3 elements
[0]: string = a
[1]: string = b
[2]: string = c
Log Object: inc
GlideRecord('incident') @ INC0000002

**JSUtil - nil(Object item)**

Checks if item is null, undefined, or evaluates to the empty string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>Object</td>
<td>The object to check</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the item is null, undefined, or evaluates to the empty string.</td>
</tr>
</tbody>
</table>

var x = "the quick brown fox";
var y = "";
var z;

gs.print("x = '" + x + "'", JSUtil.nil(x) = " + JSUtil.nil(x));
gs.print("y = '" + y + "'", JSUtil.nil(y) = " + JSUtil.nil(y));
gs.print("z = '" + z + "'", JSUtil.nil(z) = " + JSUtil.nil(z));

Output:

x = 'the quick brown fox', JSUtil.nil(x) = false
y = '', JSUtil.nil(y) = true
z = 'undefined', JSUtil.nil(z) = true

**JSUtil - notNil(Object item)**

Checks if item is null, undefined, or evaluates to the empty string.
**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>Object</td>
<td>The object to check</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the item exists and is not empty.</td>
</tr>
</tbody>
</table>

```javascript
defineProperty(ByteArray, 'notNil', {
    value: function(o) {
        if (o === null || o === undefined) {
            return false;
        }
        return o;
    }
});

var x = "the quick brown fox";
var y = "";
var z;

gs.print("x = '" + x + "', JSUtil.notNil(x) = " +
JSUtil.notNil(x));
gs.print("y = '" + y + "', JSUtil.notNil(y) = " +
JSUtil.notNil(y));
gs.print("z = '" + z + "', JSUtil.notNil(z) = " +
JSUtil.notNil(z));
```

**Output:**

```
x = 'the quick brown fox', JSUtil.notNil(x) = true
y = '', JSUtil.notNil(y) = false
z = 'undefined', JSUtil.notNil(z) = false
```

**JSUtil - toBoolean(Object item)**

Converts the specified object to a Boolean.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>Object</td>
<td>The object to convert</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If the specified object is a boolean, it is passed through. Non-zero numbers return true. Null or undefined return false. Strings return true only if exactly equal to 'true'.</td>
</tr>
</tbody>
</table>

```javascript
var zero = 0;
var one = 1;
var number = 12;
```
var trueBoolean = true;
var trueString = "true";
var otherString = "random text";

gs.print("JSUtil.toBoolean(zero) = " + JSUtil.toBoolean(zero));
gs.print("JSUtil.toBoolean(one) = " + JSUtil.toBoolean(one));
gs.print("JSUtil.toBoolean(number) = " + JSUtil.toBoolean(number));
gs.print("JSUtil.toBoolean(trueBoolean) = " + JSUtil.toBoolean(trueBoolean));
gs.print("JSUtil.toBoolean(trueString) = " + JSUtil.toBoolean(trueString));
gs.print("JSUtil.toBoolean(otherString) = " + JSUtil.toBoolean(otherString));

Output:

JSUtil.toBoolean(zero) = false
JSUtil.toBoolean(one) = true
JSUtil.toBoolean(number) = true
JSUtil.toBoolean(trueBoolean) = true
JSUtil.toBoolean(trueString) = true
JSUtil.toBoolean(otherString) = false

JSUtil - type_of(Object value)
Determines the type of the specified object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Object</td>
<td>The object to check</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The type of the specified object.</td>
</tr>
<tr>
<td></td>
<td>• ‘null’ if the given value is null or undefined</td>
</tr>
<tr>
<td></td>
<td>• ‘string’ if the given value is a primitive string or a String wrapper instance</td>
</tr>
<tr>
<td></td>
<td>• ‘number’ if the given value is a primitive number or a Number wrapper instance</td>
</tr>
<tr>
<td></td>
<td>• ‘boolean’ if the given value is a primitive boolean or a Boolean wrapper instance</td>
</tr>
<tr>
<td></td>
<td>• ‘function’ if the given value is a function</td>
</tr>
<tr>
<td></td>
<td>• ‘object’ otherwise</td>
</tr>
</tbody>
</table>

var a = ["a","b","c"];
var b = 10;
var c = new GlideRecord("incident");
var d = true;
var e;

gs.print("JSUtil.type_of(a) = " + JSUtil.type_of(a));
gs.print("JSUtil.type_of(b) = " + JSUtil.type_of(b));
gs.print("JSUtil.type_of(c) = " + JSUtil.type_of(c));
gs.print("JSUtil.type_of(d) = " + JSUtil.type_of(d));
gs.print("JSUtil.type_of(e) = " + JSUtil.type_of(e));

Output:

JSUtil.type_of(a) = object
JSUtil.type_of(b) = number
JSUtil.type_of(c) = object
JSUtil.type_of(d) = boolean
JSUtil.type_of(e) = null

**JSUtil - unescapeAttr(String text)**

Restores ampersands from escaped text.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>String</td>
<td>The text</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The text with escape characters removed.</td>
</tr>
</tbody>
</table>

```
var attr = "sysparm_query=active=true&amp;sysparm_view=special";
gs.print(JSUtil.unescapeAttr(attr));
```

Output:

sysparm_query=active=true&sysparm_view=special

**JSUtil - unescapeText(String text)**

Removes escape characters.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>String</td>
<td>The text to process.</td>
</tr>
</tbody>
</table>
var html = "&lt;b&gt;This is my title&lt;/b&gt;";

Output: This is the value returned. If the text is displayed in the application the page, it renders the html tags and displays the text in bold.

<b>This is my title</b>

Logger

Inbound email actions can use this JavaScript function to append messages to the email log. The Logger class provides methods that add standard, warning, and error messages to the email log. The added message has its source set to email.<Sys ID of incoming email>.

Logger - log(String msg)

Appends the specified message to the email log.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Text to append to the email log. These should be information type messages.</td>
</tr>
</tbody>
</table>

logger.log("Some information");

Logger - logError(String msg)

Appends the specified error message to the email log file.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Error message to append to the email log</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
logger.logError("Some error");
```

#### Logger - logWarning(String msg)

Appends the specified warning message to the email log file.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Warning message to append to the email log</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
logger.logWarning("Some warning");
```

#### LSOFParser

Parses the output of the `lsof` command: `lsof -i4TCP -n -P -F pcnfT`

Use this API with a discovery script when you need to parse `lsof` output.

#### LSOFParser - error(String msg)

Generates the specified error message.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>The error message</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**LSOFParser - initFileDescriptor()**

Initializes the file descriptor process.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**LSOFParser - initProcess()**

Initializes the parser process.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**LSOFParser - on_p(String line)**

Sets the current PID.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>String</td>
<td>The PID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
**LSOFParser - on_c(String line)**

Sets the current command.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>String</td>
<td>The command</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**LSOFParser - on_f(String line)**

Sets the current file descriptor.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>String</td>
<td>The file descriptor</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**LSOFParser - on_n(String line)**

Sets the current address.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>String</td>
<td>The address</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**LSOFParser - on_T(String line)**

Sets the current state.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>String</td>
<td>The state</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### LSOFParse - on_endFileDescriptor()

Closes the file descriptor process.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### LSOFParse - on_endProcess()

Closes the parser process.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### LSOFParse - parse(String lsofOutput)

Parses the specified lsof output.

Results are available in connections and listeners arrays, and errors are recorded in errorString() and isValid().
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsofOutput</td>
<td>String</td>
<td>The lsof output</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**MIDServer**

Encapsulates the notion of a MID server. Use these methods in server scripts to manage a MID server using Ajax.

**MIDServer - getName(String name)**

Returns the specified MID server.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The MID server name</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The record of the requested MID server.</td>
</tr>
</tbody>
</table>

**MIDServer - getForPing(DiscoverySchedule schedule, DiscoveryRange range)**

Returns the MID Server with the specified schedule and range set.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>schedule</td>
<td>DiscoverySchedule</td>
<td>The discovery schedule</td>
</tr>
<tr>
<td>range</td>
<td>DiscoveryRange</td>
<td>The discovery range set</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The MID server</td>
</tr>
</tbody>
</table>
MIDServer - getDefault(DiscoverySchedule schedule)

Returns the MID server associated with the specified schedule (the MID server with the same name). If there is no associated MID server, gets the next MID server in the ecc_agent table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>schedule</td>
<td>DiscoverySchedule</td>
<td>The discovery schedule</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The MID server</td>
</tr>
</tbody>
</table>

MIDServer - hostname

The name of the MID server's host.

Field

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostname</td>
<td>String</td>
<td>The name of the MID server's host.</td>
</tr>
</tbody>
</table>

MIDServer - hostOS

The operating system of the MID server's host.

Field

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostOS</td>
<td>String</td>
<td>The operating system of the MID server's host.</td>
</tr>
</tbody>
</table>

MIDServer - ip

The IP address of the MID server's host.

Field

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>String</td>
<td>The IP address of the MID server's host.</td>
</tr>
</tbody>
</table>

MIDServer - name

Name of the MID server.
## MIDServer - network

The network containing the MID server's host.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>network</td>
<td>String</td>
<td>The network containing the MID server's host. For example, &quot;10.10.10.0/24&quot; or &quot;10.10.10.0/255.255.255.0&quot;.</td>
</tr>
</tbody>
</table>

## MIDServer - routerIP

The IP address of the MID server host's default router.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>routerIP</td>
<td>String</td>
<td>The IP address of the MID server host's default router.</td>
</tr>
</tbody>
</table>

## MIDServer - status

The MID server's status ("Up" or "Down").

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>String</td>
<td>The MID server's status (&quot;Up&quot; or &quot;Down&quot;).</td>
</tr>
</tbody>
</table>

## MIDServer - sysID

The sys_id of the MID server record.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysID</td>
<td>String</td>
<td>The sys_id of the MID server record.</td>
</tr>
</tbody>
</table>

## MIDServer - url

The URL the MID server uses to contact the instance.
<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>String</td>
<td></td>
<td>The URL the MID server uses to contact the instance.</td>
</tr>
</tbody>
</table>

**MIDServer - version**

Version of the MID server (WAR name).

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>String</td>
<td></td>
<td>Version of the MID server (WAR name).</td>
</tr>
</tbody>
</table>

**MIDServer - windowsDomain**

Windows domain of the MID server’s host (if it is a Windows machine).

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>windowsDomain</td>
<td>String</td>
<td></td>
<td>The Windows domain of the MID server’s host.</td>
</tr>
</tbody>
</table>

**MIDServerAjax**

Provides AJAX functionality for sending a test probe to the MID server.
Use in server scripts to test a MID server using AJAX.

**MIDServerAjax - ajaxFunction_testProbe()**

Sends a test probe to the MID server.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td></td>
<td>Contains the agent name, test probe ID, topic, name, and source.</td>
</tr>
</tbody>
</table>

```javascript
var msaj = new MIDServerAjax();
```
MIDServerAjax - MIDServerAjax()

Creates an instance of MIDServerAjax.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MIDServerCluster

Finds the MID server clusters for a given MID server and reassigns the jobs if necessary.

If the MID server agent is up, this class gathers all MID servers (including the original agent) in the load balance clusters that the original agent is part of. If the MID server agent is down and the cluster is load balanced, the class gathers the other agents. If the server agent is down, and the cluster is failover, it gathers the next failover agent. The class then randomly returns the MID server in the final list of MID servers.

Use in server scripts to get MID server cluster information.

MIDServerCluster - clusterExists()

Determines if a cluster exists.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if a cluster exists; otherwise, false.</td>
</tr>
</tbody>
</table>

```
var midgr = new GlideRecord("ecc_agent");
midgr.addQuery("name", "al1winmid");
midgr.query();
midgr.next();

var mscl = new MIDServerCluster(midgr);
gs.print(mscl.clusterExists());
```

Output: true

MIDServerCluster - getClusterAgent()

Gets the cluster agent.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The agent name</td>
</tr>
</tbody>
</table>

// Assuming "mid1" and "mid2" are in a cluster
var midgr = new GlideRecord("ecc_agent");
midgr.addQuery("name", "mid1");
midgr.query();
midgr.next();

var mscl = new MIDServerCluster(midgr);
gs.print(mscl.getClusterAgent());

Output: mid2

MIDServerCluster - MIDServerCluster(GlideRecord agent)

Brief description of the method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent</td>
<td>GlideRecord</td>
<td>The MID server agent</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

MIDServerFileSync

Checks to see if there are any changes to attachments on tables that are instances of MID server synchronized files (ecc_agent_sync_file) and, if so, notifies the MID servers of a change.

The tables that extend the sync tables are MID Server JAR file (ecc_agent_jar) and MID Server MIB File (ecc_agent_mib).

Use in server scripts to notify MID servers.

MIDServerFileSync - MIDServerFileSync()

Creates an instance of MIDServerFileSync.
### MIDServerFileSync - notifyMIDServers(String table)

Checks for any changes to attachments on the specified table that are instances of MID server synchronized files and, if so, notifies the MID servers of a change.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>The table to check</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var msfs = new MIDServerFileSync();
msfs.notifyMIDServers('ex.table.name');
```

### MIDServerFinder

Finds a list of MID servers for given IP addresses.

Use in server scripts to get the server list.

**MIDServerFinder - getMIDServers()**

Gets the names of MID servers available for a given range and capability.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of MID server names. If no MID servers are available, returns an empty array.</td>
</tr>
</tbody>
</table>

```javascript
var msf = new MIDServerFinder();
msf.setRanges('10.10.10.1-10.10.11.254');
```
var msnames = msf.getMIDServers();
for(var i=0; i<msnames.length; i++) {
    gs.print(msnames[i]);
};

**MIDServerFinder - getMIDServersBySysId()**

Gets the sys_ids of MID servers available for a given range and capability.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of MID server sys_ids. If no MID servers are available, returns an empty array.</td>
</tr>
</tbody>
</table>

var msf = new MIDServerFinder();
msf.setRanges('10.10.10.1-10.10.11.254');
var msids = msf.getMIDServersBySysId();
for(var i=0; i<msids.length; i++) {
    gs.print(msids[i]);
};

**MIDServerFinder - getStatusMessage()**

Gets the state of the finder operation.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The status message</td>
</tr>
</tbody>
</table>

var msf = new MIDServerFinder();
gs.print(msf.getStatusMessage());
**MIDServerFinder - MIDServerFinder()**

Creates an instance of MIDServerFinder.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**MIDServerFinder - setActive(Boolean flag)**

Sets whether to look for active or inactive MID servers. By default, searches are for active MID servers unless inactive is specified by this method.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flag</td>
<td>Boolean</td>
<td>If true, look for inactive MID servers. If false, do not look for inactive MID servers.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var msf = new MIDServerFinder();
msf.setActive('true');
```

**MIDServerFinder - setCapabilities(Array capabilities)**

Sets the technologies for which to look.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>capabilities</td>
<td>Array</td>
<td>Contains a list of capabilities, for example: capabilities = ['ssh', 'wmi', 'snmp', {&quot;os_domain&quot;: &quot;disco&quot;}, {&quot;phase&quot;: 1}];</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var msf = new MIDServerFinder();
msf.setRanges('10.10.10.1-10.10.11.254');
var capab = ['ssh','wmi','snmp'];
msf.setCapabilities(capab);
```

**MIDServerFinder - setDebug(Boolean onOrOff)**

Turns debugging on or off.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOrOff</td>
<td>Boolean</td>
<td>True to turn on debugging; false to turn debugging off.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var msf = new MIDServerFinder();
msf.setDebug('true');
```

**MIDServerFinder - setRanges(String ranges)**

Sets the range of IP addresses for which to look.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| ranges  | String  | A comma-separated list in one of these formats.  
- IP addresses (10.10.10.1, 10.10.10.2)  
- IP networks (10.10.10.0/23)  
- IP ranges (10.10.10.1-10.10.11.254) |
MIDServerFinder - setRangesByIPorHostname(String ipOrHostname)

Determines if the input is a single IP or a hostname, and passes the IP or multiple IPs to the setRanges() method.

If the value of ipOrHostname is an IP address, it is passed into the setRanges() method. If the value is a hostname, we look up the DNS table (cmdb_ip_address_dns_name) to try resolving the hostname. The result, either an IP or multiple IPs, is then passes into the setRanges() method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipOrHostname</td>
<td>String</td>
<td>The IP address, IP range, or hostname.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

MIDServerManage

Allows you to manage a MID server, by sending messages to start, stop, upgrade, get logs, send test probes, and send system messages.

Use in server scripts to send messages to a MID server.

MIDServerManage - grab_logs(String agentName, String logs)

Gets the logs for the specified MID server.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentName</td>
<td>String</td>
<td>The MID server name</td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The log message</td>
</tr>
</tbody>
</table>

```javascript
var msm = new MIDServerManage();
msm.grab_logs('serv1');
```

Output: Background message, type:info, message: Grabbing MID Server Logs

### MIDServerManage - MIDServerManage()

Creates an instance of MIDServerManage.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### MIDServerManage - restart(String agentName)

Restarts the specified MID server.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentName</td>
<td>String</td>
<td>The MID server name</td>
</tr>
</tbody>
</table>

### MIDServerManage - stop(String agentName)

Stops the specified MID server.

```javascript
var msm = new MIDServerManage();
msm.restart('serv1');
```
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentName</td>
<td>String</td>
<td>The MID server name</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var msm = new MIDServerManage();
msm.stop('serv1');
```

### MIDServerManage - system_msg(String agentName, String cmd, String name)

Creates a system message for the specified MID server.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentName</td>
<td>String</td>
<td>The MID server name</td>
</tr>
<tr>
<td>cmd</td>
<td>String</td>
<td>The value of the probe source field</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the probe</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var msm = new MIDServerManage();
msm.test_probe('mid.server.serv1', 'command', 'probe_id');
```

### MIDServerManage - text_probe(String agentName, String probeID, String topic, String ename, String source)

Sends a test probe for the specified MID server.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentName</td>
<td>String</td>
<td>The MID server name</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>probeld</td>
<td>String</td>
<td>The ID of the probe to send</td>
</tr>
<tr>
<td>topic</td>
<td>String</td>
<td>The information for the topic field</td>
</tr>
<tr>
<td>ename</td>
<td>String</td>
<td>The name for the probe</td>
</tr>
<tr>
<td>source</td>
<td>String</td>
<td>The information for the source field</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The probe for the specified MID server</td>
</tr>
</tbody>
</table>

```javascript
var msm = new MIDServerManage();
msm.test_probe('mid.server.serv1', 'probe_id', 'topic text', 'name', 'source text');
```

**MIDServerManage - upgrade(String agentName)**

Upgrades the specified MID server.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentName</td>
<td>String</td>
<td>The MID server name</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var msm = new MIDServerManage();
msm.upgrade('serv1');
```

**MIDServerSelector**

Finds a MID server based on capabilities and target IP address.

Use in server scripts to find a MID server with specific capabilities.

**MIDServerSelector - findAgent(String target)**

Finds the MID server based on the capabilities and target IP address.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>String</td>
<td>The target IP address</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var mss = new MIDServerSelector();
var capab = ['ssh', 'wmi', 'snmp'];
mss.setCapabilities(capab);
mss.findAgent('100.101.10.10');
```

**MIDServerSelector - getError()**

Gets the error messages for the current MID server.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var mss = new MIDServerSelector();
var capab = ['ssh', 'wmi', 'snmp'];
mss.setCapabilities(capab);
mss.findAgent('100.101.10.10');
mss.getError();
```

**MIDServerSelector - getWarning()**

Gets the warning messages for the current MID server.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var mss = new MIDServerSelector();
var capab = ['ssh', 'wmi', 'snmp'];
mss.setCapabilities(capab);
mss.findAgent('100.101.10.10');
mss.getWarning();
```
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var mss = new MIDServerSelector();
var capab = "ssh","wmi","snmp";
mss.setCapabilities(capab);
mss.findAgent('100.101.10.10');
mss.getWarning();

### MIDServerSelector - MIDServerSelector()

Creates an instance of MIDServerSelector.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### MIDServerSelector - setCapabilities(Array capabilities)

Sets the technologies for which to look.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>capabilities</td>
<td>Array</td>
<td>A list of capabilities, for example - capabilities = &quot;ssh&quot;,&quot;wmi&quot;,&quot;snmp&quot;, {&quot;os_domain&quot;:&quot;disco&quot;}, {&quot;phase&quot;:1};</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var mss = new MIDServerSelector();
var capab = "ssh","wmi","snmp";
MonitorMIDServer

Monitors MID servers using the heartbeat probe to check for MID servers that are down, sends heartbeat requests, and kills old heartbeat requests.

Use in a server script to monitor MID servers.

MonitorMIDServer - killOldRequests()

If there are any heartbeat probe requests that haven’t been processed, cancels them so that they don’t accumulate when a MID server is down for a while.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var mmsv = new MonitorMIDServer();
mmsv.killOldRequests();
```

MonitorMIDServer - markDowners()

Marks any non-responding MID servers as being down, by determining when the most recent heartbeat was sent and finding all non-responding servers.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var mmsv = new MonitorMIDServer();
mmsv.markDowners();
```
MonitorMIDServer - monitor()
Performs the functions required for a scheduled MID server monitor cycle, including killing old requests, marking the MID servers that are down, and sending heartbeat requests to all servers.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
var mmsv = new MonitorMIDServer();
mmsv.monitor();
```

MonitorMIDServer - MonitorMIDServer()
Creates an instance of MonitorMIDServer.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MonitorMIDServer - sendHeartbeatRequests()
Sends heartbeat requests to all MID servers and notes when by updating the server status record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
var mmsv = new MonitorMIDServer();
mmsv.sendHeartbeatRequests();
```
NameValuePairs

Converts between maps and strings representing the name/value pairs in the map.

The string form is: `<name> = <value>, <name> = <value>, ...` where `<name>` is a string name, optionally surrounded by double quotes (Microsoft-style), and `<value>` is a string value, also optionally surrounded by double quotes.

The following are all examples of valid name/value strings:

- name = value
- name = "My Value", name = value
- "My Name = this" = "My Value", "My Value, all the time", name = value
- "My Name" = "This "name"", name = value

In the third example, the quoted values contain equals and commas. In the fourth example, the quoted value contains a quote.

Instances have the following properties initialized:

- map: the map (JavaScript object being used as a hashmap) form of the name/value pairs, with Microsoft-style quotes removed
- string: the string form of the name/value pairs, with Microsoft-style quoting.

Use with any server-side script where you need convert name/value pairs between maps and strings.

NameValuePairs - mapToString()

Produces `this.string` from `this.map`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

NameValuePairs - NameValuePairs(String stringForm)

Creates an instance of the class initialized with the specified string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stringForm</td>
<td>String</td>
<td>The values to use when creating the object.</td>
</tr>
</tbody>
</table>

NameValuePairs - NameValuePairs(Object mapForm)

Creates an instance of the class initialized with the specified hash map.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapForm</td>
<td>Object</td>
<td>The map to use when creating the object.</td>
</tr>
</tbody>
</table>

#### NameValuePairs - quotify(Object value)

If the specified value contains double quotes, equals, commas, or spaces, this method surrounds the value with double quotes and doubles up any enclosed double quotes.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Object</td>
<td>The value to process</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value with added quotes.</td>
</tr>
</tbody>
</table>

#### NameValuePairs - stringToMap()

Produce this.map from this.string.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### NetwareHandler API

Implements an SNMP handler for Netware OS.

This handler is designed to be invoked by Classify.java as an interim step toward completely moving the SNMP sensors into JavaScript.

Use this API for SNMP classification.

#### NetwareHandler - classifyAndIdentify()

Classifies and identifies the SNMP device.
### Notify

The Notify API allows you to interact with Notify calls and SMS messages using scripts.

**Notify - call(String notifyPhoneNumber, String toPhoneNumber)**

Makes a call to an E.164-compliant phone number.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyPhoneNumber</td>
<td>String</td>
<td>The phone number to make the call from. This number appears as the caller ID.</td>
</tr>
<tr>
<td>toPhoneNumber</td>
<td>String</td>
<td>The phone number to call.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var from = '+14048007337';
var to = '+31646810495';

// set up call
new SNC.Notify().call(from, to);
```

**Notify - call(String notifyPhoneNumber, String toPhoneNumber, GlideRecord conferenceCall)**

Call a number to add that number to an active conference call.

This method is intended specifically for conference calls. To initiate calls between only two participants, use the `call(String notifyPhoneNumber, String toPhoneNumber)` method instead.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyPhoneNumber</td>
<td>String</td>
<td>The Notify phone number to make the call from. When you initiate a call, the outgoing call workflow for the number group associated with this number runs. Ensure this workflow includes a <strong>join conference call</strong> activity to connect the user to the conference call.</td>
</tr>
<tr>
<td>toPhoneNumber</td>
<td>String</td>
<td>The phone number to call. Called numbers are added to the conference call.</td>
</tr>
<tr>
<td>conferenceCall</td>
<td>GlideRecord</td>
<td>A GlideRecord for the Notify Call (notify_call) table identifying the conference call record. This record is automatically added to the outgoing call workflow scratchpad as the workflow.scratchpad.conference_call variable.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var notify = new SNC.Notify();
var from = '+14041234567';
var participants = ['+31612345678', '+31623456789', '+31687654321'];

// set up a conference call
var conferenceCall = notify.conferenceCall();

// set up the outbound calls for all conference call participants
for (var i in participants) {
  var to = participants[i];
  notify.call(from, to, conferenceCall);
}

// feedback
gs.log(gs.getMessage('set up a conference call with number {0} and (re)join code: {1}',
  [ conferenceCall.getValue('number'),
    conferenceCall.getValue('code') ]));
```
**Notify - conferenceCall()**

Create a new conference call GlideRecord.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>A new Notify Call (notify_call) record for a conference call.</td>
</tr>
</tbody>
</table>

```javascript
var notify = new SNC.Notify();
var from = '+14041234567';
var participants = ['+31612345678', '+31623456789', '+31687654321'];

// set up a conference call
var conferenceCall = notify.conferenceCall();

// set up the outbound calls for all conference call participants
for (var i in participants) {
    var to = participants[i];
    notify.call(from, to, conferenceCall);
}

// feedback
gs.log(gs.getMessage('set up a conference call with number {0} and (re)join code: {1}',
    [ conferenceCall.getValue('number'),
    conferenceCall.getValue('code') ]));
```

**Notify - dequeueCall(GlideRecord callRecord)**

Resume a call after it was put in a queue.

Use this method to resume calls that were put in a queue with the queueCall method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callRecord</td>
<td>GlideRecord</td>
<td>A GlideRecord object on the Notify Call (notify_call) table with the held call you want to resume.</td>
</tr>
</tbody>
</table>
var call = new GlideRecord('notify_call');
call.set('some sys_id');
if (call.isValid()) {
    new SNC.Notify().dequeueCall(call);
}

Notify - forwardCall(GlideRecord call, String destination, String dtmf)
Forward a call to connect that call with a different recipient.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>call</td>
<td>GlideRecord or String</td>
<td>A Notify call record, or the telephony provider call ID.</td>
</tr>
<tr>
<td>destination</td>
<td>GlideRecord or String</td>
<td>A Notify phone number record, or an E.164-compliant phone number.</td>
</tr>
<tr>
<td>dtmf</td>
<td>String</td>
<td>A DTMF code to play upon connection.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Notify - getAvailableClients(String notifyNumber)
Returns a list of client sessions that are available to receive calls.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyNumber</td>
<td>String</td>
<td>Any valid Notify phone number.</td>
</tr>
</tbody>
</table>
### Notify - getChildCallIDs(GlideRecord callRecord)

Returns the sys_id values of calls that are children of a specified call.

Any call started by forwarding another call, such as with the Forward workflow activity, is considered a child of the original call. The original call is the parent call.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callRecord</td>
<td>GlideRecord</td>
<td>A record on the Notify Call (notify_call) table.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>The sys_id values of all calls that are children of the specified call.</td>
</tr>
</tbody>
</table>

```javascript
var callRecord = new GlideRecord('notify_call');
callRecord.get("0f4f5863ff13310014ecffffffff28");

var notify = new SNC.Notify();
var childCallIDs = notify.getChildCallIDs(callRecord);

for(var callID in childCallIDs)
  //perform any operations with the child callID values
```

### Notify - getParentCallID(GlideRecord callRecord)

Returns the sys_id of a specified call's parent call.

Any call started by forwarding another call, such as with the Forward workflow activity, is considered a child of the original call. The original call is the parent call.

```javascript
var callRecord = new GlideRecord('notify_call');
callRecord.get("0f4f5863ff13310014ecffffffff28");

var notify = new SNC.Notify();
var parentCallID = notify.getParentCallID(callRecord);

//perform any operations with the parent callID
```
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callRecord</td>
<td>GlideRecord</td>
<td>A record on the Notify Call (notify_call) table.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the parent call record.</td>
</tr>
</tbody>
</table>

```javascript
var callRecord = new GlideRecord('notify_call');
callRecord.get("0f4f5863ff13310014ecfffffff28");

var notify = new SNC.Notify();
var parentCallID = notify.getParentCallID(callRecord);
```

Notify - getPhoneNumbers()

Returns all phone numbers and short codes available to Notify, as an array.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of NotifyPhoneNumber objects, each object representing one phone number available to Notify.</td>
</tr>
</tbody>
</table>

```javascript
// instantiate notify
var notify = new SNC.Notify();

// get all available phone numbers
var phoneNumbers = notify.getPhoneNumbers();

// iterate over phone numbers
for (var i = 0; i < phoneNumbers.size(); i++) {
    var number = phoneNumbers.get(i);
    //perform any actions using each phone number
```
### Notify - getTokens()

*Returns client tokens for any installed telephony drivers for use in WebRTC or mobile clients.*

This function uses the currently logged-in user record as the client.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The client tokens, as a JSON string.</td>
</tr>
</tbody>
</table>

```javascript
// get Notify Client Tokens per active Notify Driver for the currently logged in user
var json = new SNC.Notify().getTokens();

// parse the json that was return into a tokens object
var tokens = JSON.parse(json);

// log line
gs.log('Notify Client Tokens for the currently logged in user');

// iterate over the driver tokens
for (var driver in tokens) {
    gs.log(driver + ' Driver token: ' + tokens[driver]);
}
```

### Notify - getTokens(GlideRecord record)

*Get client tokens for any installed telephony drivers for use in WebRTC or mobile clients.*

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>GlideRecord</td>
<td>A record used to generate the client tokens.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The client tokens. The value of the string depends on the GlideRecord parameter.</td>
</tr>
</tbody>
</table>
This example demonstrates getting Notify client tokens for every Notify group.

```javascript
// instantiate Notify
var notify = new SNC.Notify();

// get all Notify Groups
var notifyGroup = new GlideRecord("notify_group");
notifyGroup.query();

// iterate over all notify groups
while (notifyGroup.next()) {
    // generate Notify Client tokens per active Notify Driver for this group
    var json = notify.getTokens(notifyGroup);
    var tokens = JSON.parse(json);

    for (var driver in tokens) {
        gs.log(gs.getMessage("Notify Client token for {0} driver and Notify Group '{1}': {2}", [driver, notifyGroup.getValue('name'), tokens[driver]]));
    }
}
```

**Notify - kick(GlideRecord participant)**

Kicks a specified user from a Notify conference call.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>participant</td>
<td>GlideRecord</td>
<td>A GlideRecord object containing the Notify Participant (notify_participant) to kick from the conference call.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var participant = new GlideRecord('notify_participant');
participant.get('<sys_id>');
if (participant.isValid()) {
    new SNC.Notify().kick(participant);
}
```

**Notify - queueCall(GlideRecord callRecord)**

Put a call into a queue.
Resume a queued call using the dequeueCall method.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callRecord</td>
<td>GlideRecord</td>
<td>A GlideRecord object on the Notify Call (notify_call) table with the call you want to put on hold.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var call = new GlideRecord('notify_call');
call.get('<call record sys_id>');
if (call.isValid()) {
    new SNC.Notify().queueCall(call);
}
```

### Notify - sendSMS(NotifyPhoneNumber notifyPhoneNumber, String toPhoneNumber, String messageBody)

Sends an SMS text message to an E.164-compliant phone number.

This function creates a new record on the Notify Message (notify_message) table.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyPhoneNumber</td>
<td>NotifyPhoneNumber</td>
<td>The Notify phone number or short code to send this SMS from.</td>
</tr>
<tr>
<td>toPhoneNumber</td>
<td>String</td>
<td>An E.164-compliant phone number to send the SMS to.</td>
</tr>
<tr>
<td>messageBody</td>
<td>String</td>
<td>The SMS text.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the Notify Message (notify_message) record created by this function.</td>
</tr>
</tbody>
</table>

### Notify - sendSMS(NotifyPhoneNumber notifyPhoneNumber, String toPhoneNumber, String messageBody, GlideRecord source)

Sends an SMS text message to an E.164-compliant phone number.
This function creates a new record on the Notify Message (notify_message) table and associates it with the source record.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyPhoneNumber</td>
<td>NotifyPhoneNumber</td>
<td>The Notify phone number or short code to send this SMS from.</td>
</tr>
<tr>
<td>toPhoneNumber</td>
<td>String</td>
<td>An E.164-compliant phone number to send the SMS to.</td>
</tr>
<tr>
<td>messageBody</td>
<td>String</td>
<td>The SMS text.</td>
</tr>
<tr>
<td>source</td>
<td>GlideRecord</td>
<td>The source record that prompted this SMS message, such as an incident.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the Notify Message (notify_message) record created by this function.</td>
</tr>
</tbody>
</table>

### NotifyAction

The NotifyAction API allows you to define actions to send to a telephony provider.

You add actions to a NotifyAction object by calling the respective add function for each type of action.

Each add function returns an Action object, such as a SayAction object for the `addSay()` function. Refer to each method example for information about returned objects.

#### NotifyAction - addConference()

Add a call to a Notify conference call.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConferenceAction</td>
<td>The action added to the NotifyAction object. Use the ConferenceAction object to define the conference call name, and the behavior of the conference call when a participant joins or leaves.</td>
</tr>
</tbody>
</table>
**NotifyAction - addDial()**

Makes an outbound call.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DialAction</td>
<td>The action added to the NotifyAction object. Use the DialAction object to define.</td>
</tr>
</tbody>
</table>

```javascript
var action = new SNC.NotifyAction();
var dial = action.addDial();
dial.setRecord(activity.vars.record);
dial.setClientRecord(activity.vars.user, "sys_user");
```

**NotifyAction - addGather()**

Presents an interactive phone menu to the user.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GatherAction</td>
<td>The action added to the NotifyAction object. Use the GatherAction object to define the menu settings and options to present to the user.</td>
</tr>
</tbody>
</table>

```javascript
// instantiate NotifyAction
var notifyAction = new SNC.NotifyAction();

// present the user with a menu
var gather = notifyAction.addGather();
gather.setNumberOfDigits(1);    // the user can type 1 digit
gather.setFinishKey('#');      // # or *, usefull for > 1 digits
gather.setTimeout(10);          // time to enter answer, in seconds
```
// add first menu item
var usSay = gather.addSay();
usSay.setText('Press 1 for english');
usSay.setLanguage('en-US');

// add second menu item
var nlSay = gather.addSay();
nlSay.setText('Kies 2 voor Nederlands');
nlSay.setLanguage('nl-NL');

// add third menu item
var frSay = gather.addSay();
frSay.setText('Choisissez 3 pour le français.');
frSay.setLanguage('fr-FR');

// and finish off with an applause
var play = gather.addPlay();
play.setURL('http://www.wavsource.com/snds_2015-04-12_5971820382841326/sfx/applause_y.wav');

**NotifyAction - addHangUp()**

Ends an active phone call.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HangUpAction</td>
<td>The action added to the NotifyAction object.</td>
</tr>
</tbody>
</table>

// instantiate NotifyAction
var notifyAction = new SNC.NotifyAction();

// hang up
notifyAction.addHangUp();

**NotifyAction - addQueue()**

Queue the call, putting it on hold.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueueAction</td>
<td>The action added to the NotifyAction object. Use the QueueAction object to define the queue name, and queueing or dequeueing behavior.</td>
</tr>
</tbody>
</table>

```javascript
// instantiate NotifyAction
var notifyAction = new SNC.NotifyAction();

// queue the call
var queue = notifyAction.addQueue();
queue.setName('my queue');
```

```javascript
// instantiate NotifyAction
var notifyAction = new SNC.NotifyAction();

// dequeue the call
var queue = notifyAction.addQueue();
queue.setDequeue(true);
```

**NotifyAction - addPlay()**

Plays an audio file on the call.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayAction</td>
<td>The action added to the NotifyAction object. Use the PlayAction object to define the audio file URL and number of times to loop the audio.</td>
</tr>
</tbody>
</table>

```javascript
// instantiate NotifyAction
var notifyAction = new SNC.NotifyAction();

// add a play action
var play = notifyAction.addPlay();
play.setURL('http://www.moviesounds.com/2001/imsorry.wav');
play.setLoop(1);```
NotifyAction - addReject()

Rejects an incoming call.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RejectAction</td>
<td>The action added to the NotifyAction object. Use the RejectAction object to define the reason for rejecting the call.</td>
</tr>
</tbody>
</table>

```javascript
// instantiate NotifyAction
var notifyAction = new SNC.NotifyAction();

// reject the call
var rejectAction = notifyAction.addReject();
rejectAction.setReason('busy'); // 'busy' or 'rejected'
```

NotifyAction - addSay()

Use text-to-speech to read text on the call.

Multiple languages are supported with text-to-speech. Available languages depend on the telephony provider.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SayAction</td>
<td>The action added to the NotifyAction object. Use the SayAction object to define the text and language to read.</td>
</tr>
</tbody>
</table>

This example demonstrates reading text in several languages.

```javascript
// instantiate NotifyAction
var notifyAction = new SNC.NotifyAction();

// add a say action to say something in US English
```
```javascript
var usSay = notifyAction.addSay();
usSay.setText('Welcome. I can speak english');
usSay.setLanguage('en-US');

// add a say action to say something in Dutch
var nlSay = notifyAction.addSay();
nlSay.setText('Ik spreek ook vloeiend nederlands');
nlSay.setLanguage('nl-NL');

// and german
var deSay = notifyAction.addSay();
deSay.setText('Und ich kann auch deutsch sprechen');
deSay.setLanguage('de-DE');
```

**NotifyAction - addSMS()**

Sends an SMS message.

When using this function with an active call, you do not need to call the setTo function on the returned SMSAction object. The SMS is automatically sent to the caller.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMSAction</td>
<td>The action added to the NotifyAction object. Use the SMSAction object to define the message text and the phone number to send the message to.</td>
</tr>
</tbody>
</table>

```javascript
// instantiate NotifyAction
var notifyAction = new SNC.NotifyAction();

// define where to send the sms to
var number = new GlideElementPhoneNumber();
number.setPhoneNumber('+31612345678', true);

// add a SMS action
var sms = notifyAction.addSMS();
sms.setMessage('Lorem ipsum dolor sit amet, consectetur adipiscing elit.');
sms.setTo(number);
```

**NotifyAction - fromJson(String json)**

Deserializes a NotifyAction object from a JSON string.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>json</td>
<td>String</td>
<td>A JSON string representation of a NotifyAction object.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

This example demonstrates deserializing a NotifyAction object.

```javascript
var json = ".... some json obtained from toJson ....";

// instantiate notify action
var notifyAction = new SNC.NotifyAction();

// deserialize and reconstruct the notify action instance
notifyAction.fromJson(json);
```

This example demonstrates both serializing and deserializing a NotifyAction object.

```javascript
// instantiate notify action
var notifyAction = new SNC.NotifyAction();

// add a queue
var queue = notifyAction.addQueue();
queue.setName('myQueueName');
queue.setDequeue(false);

// serialize to json
var json = notifyAction.toJson();
gs.log('serialization result: ' + json);

// instantiate a new notify action
var newAction = new SNC.NotifyAction();

// deserialize the json generated above
newAction.fromJson(json);

// serialize the new object and log the result
newJson = newAction.toJson();
gs.log('new serialization result: ' + newJson);
gs.log('the same: ' + (json == newJson));
```

Output: *** Script: serialization result: {"fClassName":"NotifyAction","fActions":[{"fClassName":"QueueAction","fDequeue":true,"fQueueName":"myQueueName"}]}

*** Script: new serialization result: {"fClassName":"NotifyAction","fActions":[{"fClassName":"QueueAction","fDequeue":true,"fQueueName":"myQueueName"}]}

*** Script: the same: true
NotifyAction - toJson()

Serialize the NotifyAction object to a JSON string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON representation of this NotifyAction object.</td>
</tr>
</tbody>
</table>

This example demonstrates serializing a NotifyAction object.

```javascript
// instantiate notify action
var notifyAction = new SNC.NotifyAction();

// add one or more notify actions
// ...

// and serialize to json
var json = notifyAction.toJson();
```

This example demonstrates both serializing and deserializing a NotifyAction object.

```javascript
// instantiate notify action
var notifyAction = new SNC.NotifyAction();

// add a queue
var queue = notifyAction.addQueue();
queue.setName('myQueueName');
queue.setDequeue(false);

// serialize to json
var json = notifyAction.toJson();
gs.log('serialization result: ' + json);

// instantiate a new notify action
var newAction = new SNC.NotifyAction();

// deserialize the json generated above
newAction.fromJson(json);

// serialize the new object and log the result
newJson = newAction.toJson();
gs.log('new serialization result: ' + newJson);
gs.log('the same: ' + (json == newJson));
```

Output: *** Script: serialization result: {"fClassName":"NotifyAction","fActions":null} *** Script: new serialization result: {"fClassName":"NotifyAction","fActions":null}
NotifyPhoneNumber

The NotifyPhoneNumber API allows you to query information about a Notify phone number.

NotifyPhoneNumber - getDialCode()

Returns the international dialing code for a Notify phone number.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>International phone code for a country.</td>
</tr>
</tbody>
</table>

NotifyPhoneNumber - getID()

Returns the ID of this phone number as defined by the telephony provider.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The ID of the number within the telephony provider.</td>
</tr>
</tbody>
</table>

NotifyPhoneNumber - getNumber()

Returns the numerical phone number for a NotifyPhoneNumber.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### NotifyPhoneNumber - getOwner()

Returns the telephony provider associated with this phone number.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>The telephony provider associated with the number: Twilio.</td>
</tr>
</tbody>
</table>

### NotifyPhoneNumber - getTerritory()

Returns the country associated with the phone number.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The name of the country the phone number belongs to.</td>
</tr>
</tbody>
</table>

### NotifyPhoneNumber - supportsConferenceCall()

Determines if the Notify phone number supports conference calling.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Returns true if the Notify phone number supports conference calling.</td>
</tr>
</tbody>
</table>

**NotifyPhoneNumber - supportsIncomingPhoneCall()**

Determines if the Notify phone number supports receiving phone calls.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Returns true if the Notify phone number supports receiving incoming phone calls.</td>
</tr>
</tbody>
</table>

**NotifyPhoneNumber - supportsIncomingSMS()**

Determines if the Notify phone number supports receiving SMS messages.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Returns true if the Notify phone number supports receiving incoming MMS messages.</td>
</tr>
</tbody>
</table>

**NotifyPhoneNumber - supportsOutgoingPhoneCall()**

Determines if the Notify phone number supports initiating phone calls.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### NotifyPhoneNumber - supportsOutgoingSMS()

Determines if the Notify phone number supports initiating outgoing phone calls.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>boolean</td>
<td>Returns true if the Notify phone number supports initiating outgoing phone calls.</td>
</tr>
</tbody>
</table>

### NotifyPhoneNumber - supportsRecording()

Determines if the Notify phone number supports recording phone calls.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>boolean</td>
<td>Returns true if the Notify phone number supports recording phone calls.</td>
</tr>
</tbody>
</table>

### NotifyPhoneNumber - supportsWebRTC()

Determines if the Notify phone number supports calls to a browser, such as in a WebRTC implementation.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>boolean</td>
<td>Returns true if the Notify phone number supports calls to a browser, such as in a WebRTC implementation.</td>
</tr>
</tbody>
</table>
### Notify Client

The Notify Client API allows you to use Notify telephony functionality, such as making and receiving calls, from a web browser.

Several Notify Client methods take a callback function as a parameter. Because Notify Client calls are made asynchronously, these methods cannot return a value directly. Use the callback function to parse the returned data, such as by assigning variables or making other API calls.

### Notify Client - call(Object identifier)

Call a specified phone number or the phone number associated with a specified user.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier</td>
<td>Object</td>
<td>Enter a JSON object that provides either the phone number to call or the sys_id of a user record to get the phone number from.</td>
</tr>
</tbody>
</table>

**Note:** If you provide both a phone number and user sys_id, only the phone number is used.

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

This example demonstrates passing a phone number as the function parameter.

```javascript
notifyClient.call({
    phoneNumber: "+18001112223"
});
```

This example demonstrates passing a user record sys_id as the function parameter.

```javascript
notifyClient.call({
    userId: "6816f79cc0a8016401c5a33be04be441"
});
```
### Notify Client - Client(Object notifyConfig)

Instantiates a new Notify WebRTC Client object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyConfig</td>
<td>Object</td>
<td>The configuration settings for the Notify WebRTC Client, as a JSON object.</td>
</tr>
</tbody>
</table>

```javascript
var notifyConfig = {
  vendor: SNC.Notify.Vendor.TWILIO,
  callerId: 'xyz',
  autoReconnect: true,
  onReady: function() {},
  onOffline: function() {},
  onError: function( message ) {},
  onConnect: function( status ) {},
  onDisconnect: function() {},
  onIncoming: function( {from, to, callSid} ) {},
  onOutgoing: function( callSid ) {},
  onAccept: function() {},
  onMute: function() {},
  onUnmute: function() {},
  onCancel: function() {}
};

$\(function() {
  notifyClient = new SNC.Notify.Client(notifyConfig);
  notifyClient.init();
});
```

### NotifyClient - forwardCall(Object argument)

Forward the current call to a different phone number or Notify client session.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>argument</td>
<td>Object</td>
<td>A JavaScript object detailing the number or Notify client to forward the call to.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
This example demonstrates forwarding a call to a different phone number. The \textit{dtmf} attribute allows you to send DTMF dial tones to the receiving number.

```javascript
var arg = {
    type: "number",
    id: "+17012345678",
    dtmf: "1234"
}
client.forwardCall(arg);
```

This example demonstrates forwarding a call to a different Notify client.

```javascript
var arg = {
    type: "userId",
    id: "6816f79cc0a8016401c5a33be04be441"
}
client.forwardCall(arg);
```

\textbf{Notify Client - \texttt{hangupCall()}}

End the current call.

\textbf{Parameters}

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\textbf{Returns}

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

This example demonstrates mapping a client function to an interface button using jQuery.

```javascript
$\!\!j("#pickupCallBtn").on("click", function() {
    notifyClient.hangupCall();
});
```

\textbf{Notify Client - \texttt{mute(Boolean muted)}}

Mutes or unmutes the current client.

\textbf{Parameters}

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>muted</td>
<td>Boolean</td>
<td>Set this value to true to mute the current client, false to unmute.</td>
</tr>
</tbody>
</table>
### Notify Client - pickupCall()

Answers and connects to an incoming call from an Twilio WebRTC client.

Call this method when there is a notification of an incoming call.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### NotifyClient - sendDtmf(String digits)

Send one or more DTMF-valid digits over the current call.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>digits</td>
<td>String</td>
<td>One or more DTMF-valid digits.</td>
</tr>
</tbody>
</table>

### Legacy Notify API

The legacy Notify API provides functionality for sending emails, sending SMS messages, and setting up conference calls.

Use this when you want to use Notify functionality with applications on your system.

**Note:** This API is included with the legacy Notify functionality. For APIs included in the current Notify feature, see the Notify, NotifyAction, NotifyPhoneNumber, and NotifyClient APIs.

### NotifyNow - addConferenceCallParticipant(String conferenceCall, String participant)

Adds ad-hoc users to an ongoing conference call.
When the method is called with a phone number for the participant parameter and there is exactly one sys_user record that matches the phone number, that sys_user record will be related to the participant. The participant's phone number field will be left blank because the phone number is in the sys_user record. If there are several sys_user records that match the phone number, or if there are no results, the participant's phone number field will be filled in, and there will be no stored reference to sys_user because the user is not known.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conferenceCall</td>
<td>String or GlideRecord</td>
<td>The sys_id or GlideRecord of an active conference call.</td>
</tr>
<tr>
<td>participant</td>
<td>String or GlideRecord</td>
<td>The sys_id or GlideRecord of a user with an E.164-compliant phone number, or an E.164-compliant phone number.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The participant record of the new participant that was added to the conference call.</td>
</tr>
</tbody>
</table>

```javascript
// add a new participant by conference call sys_id (string) and phone number (string)
var nn = new SNC.NotifyNow();
gs.log(nn.addConferenceCallParticipant('d193b242eb020100a04d4910f206fe39', '+31612345678'));
```

```javascript
// add a new participant by conference call sys_id (string) and user record (GlideRecord)
var user = new GlideRecord('sys_user');
user.query('user_name', 'myUserName);
if (user.hasNext() & user.next()) {
    var nn = new SNC.NotifyNow();
    gs.log(nn.addConferenceCallParticipant('d193b242eb020100a04d4910f206fe39', user));
    // you could have added the user by sys_id as well:
    // nn.addConferenceCallParticipant('d193b242eb020100a04d4910f206fe39', user.getValue('sys_id'));
} else {
    gs.log('no such user');
}
```
// add a new participant by conference call record (GlideRecord)
and phone number (string)
var conferenceCall = new GlideRecord('notifynow_conference_call');
conferenceCall.query('title', 'IA0001001');
if (conferenceCall.hasNext() && conferenceCall.next()) {
  var nn = new SNC.NotifyNow();
  gs.log(nn.addConferenceCallParticipant(conferenceCall, '+31612345678'));
} else {
  gs.log('no such conference call');
}

**NotifyNow - getReadyState()**

Indicates whether Notify is set up correctly or not.

This method can only be accessed by administrators or users with the notifynow_admin role. Users with all other roles get the message False when trying to run the function in a script.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if Notify is set up correctly, otherwise false.</td>
</tr>
</tbody>
</table>

var nn = new SNC.NotifyNow();
gs.log(((nn.getReadyState()) ? "OK" : "NOT OK"));

**NotifyNow - getStatus()**

Return the current status of Notify configuration.

This method can only be accessed by administrators or users with the notifynow_admin role. Users with all other roles get the message Unauthorized when trying to run the function in a script.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>One of the possible status messages.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_NUMBER_MESSAGE</td>
<td>The account does not have a telephone number set up. Ensure that you set up the telephone number for the account.</td>
</tr>
<tr>
<td>NO_ENDPOINTS_MESSAGE</td>
<td>The account does not have its endpoints set up correctly. Ensure that you set up the endpoints for the account.</td>
</tr>
<tr>
<td>ACCOUNT_OK_MESSAGE</td>
<td>The account is active and ready for use.</td>
</tr>
<tr>
<td>ACCOUNT_NO_AUTH</td>
<td>The Twilio AuthToken is not valid.</td>
</tr>
<tr>
<td>ACCOUNT_NOT_CONFIGURED</td>
<td>Twilio AccountSID or AuthToken is not valid.</td>
</tr>
</tbody>
</table>

```javascript
var nn = new SNC.NotifyNow();
gs.log(nn.getStatus());
```

**NotifyNow - initiateConferenceCall(String[] conferenceCallParticipants, String conferenceCallTitle)**

Initiate a new conference call.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conferenceCallParticipants</td>
<td>String</td>
<td>One or more users, conference call participants, identified by the sys_ids from the sys_user table or E.164-compliant phone numbers.</td>
</tr>
<tr>
<td>conferenceCallTitle</td>
<td>String</td>
<td>Title of the conference call. This parameter has a maximum length of 40 characters.</td>
</tr>
</tbody>
</table>
This initiates a conference call with E.164-compliant phone numbers for participants, without the optional source record parameter and and does not send any conference call details via SMS or email.

```javascript
var participants = ['+31205655548', '+31205655552', '+31652825393'];
// set up conference call
var nn = new SNC.NotifyNow();
var conferenceCall = nn.initiateConferenceCall(participants, "testing12");
gs.log('started conference call: ' + conferenceCall.getUniqueValue());
```

### NotifyNow - initiateConferenceCall(String[] conferenceCallParticipants, String conferenceCallTitle, GlideRecord sourceRecord, Boolean private)

Initiate a new conference call.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conferenceCallParticipants</td>
<td>String</td>
<td>One or more users, conference call participants, identified by the sys_ids from the sys_user table or E.164-compliant phone numbers.</td>
</tr>
<tr>
<td>conferenceCallTitle</td>
<td>String</td>
<td>Title of the conference call. This parameter has a maximum length of 40 characters.</td>
</tr>
<tr>
<td>sourceRecord</td>
<td>GlideRecord</td>
<td>Source record to associate to the conference call such as an incident or problem number.</td>
</tr>
<tr>
<td>private</td>
<td>Boolean</td>
<td>Value to control if a conference call is private. This value defaults to false.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The conference call record, or null if there was an error.</td>
</tr>
</tbody>
</table>
This initiates a conference call with participants that have a E.164-compliant phone number and participants from the sys_user table and sends conference call details via SMS and email to all participants.

```javascript
// define phone number participants
var participants = ['+31205655548', '+31205655552', '+31652825393'];

// we also want to add two Dutch sys_user participants
var user = new GlideRecord('sys_user');
user.addNotNullQuery('mobile_phone');
user.addQuery('mobile_phone', 'STARTSWITH', '+316');
user.setLimit(2);
user.query();

while (user.hasNext() && user.next()) {
  gs.log('adding user ' + user.getValue('name') + ' with phone number ' +
      user.getValue('mobile_phone') + ' to the participant array');
  participants.push(user.getUniqueValue());
}

// define a source record to associate with the conference call
var source = new GlideRecord("cmdb_ci");
source.query("asset_tag", "P1000167");
if (source.hasNext() && source.next()) {
  // set up conference call
  var nn = new SNC.NotifyNow();
  var conferenceCall = nn.initiateConferenceCall(participants, "testing 1 2", source);
  // check if the conference call was successfully created
  if (conferenceCall != null) {
    gs.log('started conference call: ' +
        conferenceCall.getUniqueValue());
  } else {
    gs.log('could not start the conference call :(');
  }
}
```

**NotifyNow - isCallable(String participant)**

Determines whether a user is callable or not.

A user must have a valid phone number to be callable. A user who is already in an active session is not callable.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>participant</td>
<td>String or GlideRecord</td>
<td>A sys_user or notifynow_participant record, or an E.164-compliant phone number.</td>
</tr>
</tbody>
</table>
### NotifyNow - isSMSCapable()

Checks if the telephone number associated with the Twilio account is capable of sending SMS messages.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Whether the telephone number associated with the Twilio account is capable of sending SMS messages.</td>
</tr>
</tbody>
</table>

```javascript
var nn = new SNC.NotifyNow();
gs.log('by number: ' + nn.isCallable('+31612345678'));

var user = GlideRecord('sys_user');
user.query('sys_id', '13d39544eb5201003cf587b9d106fea9');
if (user.hasNext() && user.next())
gs.log('by user: ' + nn.isCallable(user));

var participant = GlideRecord('notifynow_participant');
participant.query('sys_id', '33b11430eb1201003cf587b9d106feb9');
if (participant.hasNext() && participant.next())
gs.log('by participant: ' + nn.isCallable(participant));
```

```javascript
gs.log('The twilio number is SMS capable: ' + ((new SNC.NotifyNow().isSMSCapable()) ? 'yes' : 'no'));
```

### NotifyNow - isSMSCapable(String userID)

Checks if a user is able to send SMS messages.
**ServiceNow**  
**Kingston**  
**Now Platform Custom Business Applications**

### NotifyNow - isSMSCapable()

Checks if the user can send SMS messages.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userID</td>
<td>String</td>
<td>The sys_id of the user you want to check for an SMS-capable phone number.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If the user can send SMS messages.</td>
</tr>
</tbody>
</table>

```javascript
gs.log('the user is able to send SMS messages (e.g. has a SMS device): ' + ((new SNC.NotifyNow().isSMSCapable('<user sys_id>')) ? 'yes' : 'no'));
```

### NotifyNow - isVoiceCapable()

Checks if the telephone number associated with the Twilio account is capable of setting up phone calls.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Whether the telephone number associated with the Twilio account is capable of setting up phone calls.</td>
</tr>
</tbody>
</table>

```javascript
gs.log('the Twilio number is Voice capable: ' + ((new SNC.NotifyNow().isVoiceCapable()) ? 'yes' : 'no'));
```

### NotifyNow - isVoiceCapable(String userID)

Checks if a user is able to make voice calls.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userID</td>
<td>String</td>
<td>The sys_id of the user you want to check for a voice-call capable phone number.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Whether the user has a voice-call capable phone number.</td>
</tr>
</tbody>
</table>

```
gs.log('the user is able to send SMS messages (e.g. has a SMS device): ' +
   ((new SNC.NotifyNow().isVoiceCapable('someuserid')) ?
    'yes' : 'no'));
```

**NotifyNow - kick(GlideRecord participant)**

Removes a participant from a conference call.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>participant</td>
<td>GlideRecord</td>
<td>The conference call participant to remove from the call.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the participant was removed, otherwise false.</td>
</tr>
</tbody>
</table>

```
var participantId = "<participant sys_id>";
var participant = new GlideRecord('notifynow_participant');
participant.get(participantId);
if (participant.isValid()) {
    // kick participant
    result = new SNC.NotifyNow().kick(participant);
    gs.log('participant kicked: ' + result);
}
```

**NotifyNow - mute(GlideRecord participant)**

Mutes a participant on a conference call.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>participant</td>
<td>GlideRecord</td>
<td>The conference call participant to mute.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the participant was muted, otherwise false.</td>
</tr>
</tbody>
</table>

```javascript
var participantId = "<participant sys_id>";
var participant = new GlideRecord('notifynow_participant');
participant.get(participantId);
if (participant.isValid()) {
    // mute participant
    result = new SNC.NotifyNow().mute(participant);
    gs.log('participant muted: ' + result);
}
```

### NotifyNow - umute(GlideRecord participant)

Unmutes a participant on a conference call.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>participant</td>
<td>GlideRecord</td>
<td>The muted conference call participant to unmute.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the participant was unmuted, otherwise false.</td>
</tr>
</tbody>
</table>

```javascript
var participantId = "<participant sys_id>";
var participant = new GlideRecord('notifynow_participant');
participant.get(participantId);
if (participant.isValid()) {
    // unmute participant
    result = new SNC.NotifyNow().unmute(participant);
    gs.log('participant unmuted: ' + result);
}
```
NotifyNow - sendEmailQuestion(String emailAddress, String question, GlideRecord sourceRecord, String emailSubject)

Send an email question to an email address.
The sendEmailQuestion method produces a question body and requires users to click a link to indicate their choice.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>String</td>
<td>Email address to send the question to.</td>
</tr>
<tr>
<td>question</td>
<td>String or GlideRecord</td>
<td>The question record to send or the sys_id of a question record.</td>
</tr>
<tr>
<td>sourceRecord</td>
<td>GlideRecord</td>
<td>An optional source record to associate to the SMS question, such as an incident.</td>
</tr>
<tr>
<td>emailSubject</td>
<td>String</td>
<td>Optional text to override the default email subject.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The conversation sys_id.</td>
</tr>
</tbody>
</table>

This example demonstrates using the default email subject.

```javascript
var user = GlideRecord("sys_user");
user.get("email", "someone@somedomain.com");
new SNC.NotifyNow().sendEmailQuestion(user.getValue('email'), "b6b34500bf3111003cf585ce2c0739ce", user);
```

This example uses dot-walking and specifies a source record and email subject.

```javascript
new SNC.NotifyNow().sendEmailQuestion("someone@somedomain.com", "b6071733bf1111003cf585ce2c07390f", current, "Please answer this question");
```

This example uses dot-walking and specifies an email subject but no source record.

```javascript
new SNC.NotifyNow().sendEmailQuestion("someone@somedomain.com", "b6071733bf1111003cf585ce2c07390f", "Please answer this question");
```
**NotifyNow - sendSMS(String phoneNumber, String smsBody)**

Sends an SMS message to an E.164-compliant mobile phone number.

Notify supports international numbers. Using this method with a number that does not support sending SMS messages results in an error being logged.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phoneNumber</td>
<td>String</td>
<td>The E.164-compliant phone number to send the message to.</td>
</tr>
<tr>
<td>smsBody</td>
<td>String</td>
<td>The message to send, maximum 1600 characters.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
new SNC.NotifyNow().sendSMS("+31612345678", "This is a message without source record");
```

**NotifyNow - sendSMS(String phoneNumber, String smsBody, GlideRecord source)**

Sends an SMS message to an E.164-compliant mobile phone number.

Notify supports international numbers. Using this method with a number that does not support sending SMS messages results in an error being logged.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phoneNumber</td>
<td>String</td>
<td>The E.164-compliant phone number to send the message to.</td>
</tr>
<tr>
<td>smsBody</td>
<td>String</td>
<td>The message to send, maximum 1600 characters.</td>
</tr>
<tr>
<td>source</td>
<td>GlideRecord</td>
<td>The source record to associate with this SMS message.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
```javascript
var source = new GlideRecord("my_table");
source.query("my_field", "my_value");

if (source.hasNext() && source.next()) {
    // send a text message
    var nn = new SNC.NotifyNow();
    var message = "this is just a test";
    var number = "+31612345678";
    nn.sendSMS(number, message, source);
}
```

This example uses dot-walking and the current record as the source record.

```javascript
new SNC.NotifyNow().sendSMS("+31612345678", "this is a test", current);
```

**NotifyNow - sendSMSQuestion(String phoneNumber, String question, GlideRecord sourceRecord)**

Sends an SMS question.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phoneNumber</td>
<td>An E.164-compliant phone number to send the message to.</td>
<td></td>
</tr>
<tr>
<td>question</td>
<td>String or GlideRecord</td>
<td>The question record to send or the sys_id of a question record.</td>
</tr>
<tr>
<td>sourceRecord</td>
<td>An optional source record to associate to the SMS question, such as an incident.</td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The conversation sys_id, or null if the SMS was not sent successfully.</td>
</tr>
</tbody>
</table>

```javascript
var question = new GlideRecord("notifynow_question");
question.query();

// get the first question
if (question.hasNext() && question.next()) {
    // send the sms question
    var number = "+31612345678";
    var nn = new SNC.NotifyNow();
    nn.sendSMSQuestion(number, question.getUniqueValue(), current);
```
NotifyNow - convertLocalPhoneNumberToE164(String userID, String phoneNumber)
Converts a local phone number to an E.164-compliant phone number based on a user's location.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userID</td>
<td>String</td>
<td>The sys_id of a sys_user record to get location information from.</td>
</tr>
<tr>
<td>phoneNumber</td>
<td>String</td>
<td>The phone number.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The E.164-compliant phone number.</td>
</tr>
</tbody>
</table>

```java
var localPhoneNumber = '01784 221600';
var userName = 'Heath Vanalphen';

var user = new GlideRecord('sys_user');
user.get('name',userName);
var E164Number = new SNC.NotifyNow().convertLocalPhoneNumberToE164(user.getUniqueValue(),
localPhoneNumber);
gs.log('converted: ' + localPhoneNumber + ' to ' + E164Number + ' based on ' + user.getValue('name') + '
' + user.getValue('location') + ')');
```

NotifyNow - getConferenceCallParticipants(String conferenceCallId, Boolean isCallable)
Returns all participants for a conference call.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conferenceCallId</td>
<td>String</td>
<td>The ID of the conference call.</td>
</tr>
<tr>
<td>isCallable</td>
<td>Boolean</td>
<td>An optional flag to return either only the users you can call (true) or those you cannot call (false).</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The participants</td>
</tr>
</tbody>
</table>
var nn = new SNC.NotifyNow();
var user = 
    nn.getConferenceCallParticipants('c2e91710eb120100f34087b9d106fe37');

while (user.hasNext() && user.next()) {
    if (user.getValue('participant')) {
        gs.log('user: ' + user.getValue('sys_id'));
    } else {
        gs.log('phone number: ' + user.getValue('phone_number'));
    }
}

var nn = new SNC.NotifyNow();
var user = 
    nn.getConferenceCallParticipants('c2e91710eb120100f34087b9d106fe37', true);

while (user.hasNext() && user.next()) {
    if (user.getValue('participant')) {
        gs.log('user: ' + user.getValue('sys_id'));
    } else {
        gs.log('phone number: ' + user.getValue('phone_number'));
    }
}

var conferenceCallId = '32b11430eb1201003cf587b9d106feb8';

// get all participants
gs.log('all conference call participants:');
var nn = new SNC.NotifyNow();
var user = nn.getConferenceCallParticipants(conferenceCallId);
gs.log(user);

// get all callable participants
gs.log('all conference call participants we can call:');
user = nn.getConferenceCallParticipants(conferenceCallId, true);
gs.log(user);

// get all un callable participants
gs.log('all conference call participants that are already in an active session and whom we cannot call:');
user = nn.getConferenceCallParticipants(conferenceCallId, false);
gs.log(user);

**NotifyNow - getFrequentlyCalledUsers(Number limit)**

Returns a number of frequently-called users, up to the limit parameter, in alphabetical order.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>Number</td>
<td>The maximum number of results.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The frequently called users in alphabetical order.</td>
</tr>
</tbody>
</table>

```javascript
var nn = new SNC.NotifyNow();
var fc = nn.getFrequentlyCalledUsers(10);

while (fc.hasNext() && fc.next()) {
    gs.log("got user " + fc.getValue('name') + ' - ' +
            fc.getValue('sys_id'));
}
```

**NotifyNow - getPreferredE164SMSNumber(GlideRecord user)**

Returns a user's preferred E.164-compliant phone number for SMS messages.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>GlideRecord or String</td>
<td>The user record or the sys_id of a user to get the E.164-compliant phone number from.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The E.164-compliant phone number or null.</td>
</tr>
</tbody>
</table>

```javascript
var userID = "<user sys_id>";
var E164Number = new SNC.NotifyNow().getPreferredE164SMSNumber(userID);
gs.log('the preferred phone number for sending SMS notifications is ' + E164Number + ' for user with id: ' + userID);
```

**NotifyNow - getPreferredE164VoiceNumber(GlideRecord user)**

Returns a user's preferred E.164-compliant phone number for voice calls.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>GlideRecord or String</td>
<td>The user record or the sys_id of a user to get the E.164-compliant phone number from.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The E.164-compliant phone number or null.</td>
</tr>
</tbody>
</table>

```javascript
var userID = "<user sys_id>";
var E164Number = new SNC.NotifyNow().getPreferredE164VoiceNumber(userID);
gs.log('the preferred phone number for setting up voice calls is ' + E164Number + ' for user with id: ' + userID);
```

**NotifyNow - getPreferredEmailAddress(GlideRecord user)**

Returns a user's preferred email address

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>GlideRecord or String</td>
<td>The user record or the sys_id of a user to get the email address from.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The email address or null.</td>
</tr>
</tbody>
</table>

```javascript
var userID = "some user sys id";
var email = new SNC.NotifyNow().getPreferredEmailAddress(userID);
gs.log('the preferred email address for sending email notifications is ' + email + ' for user with id: ' + userID);
```

**openFrameAPI**

OpenFrame is an omni-present frame that communication partners can use to integrate their systems into the ServiceNow platform.
One of the core requirements is the ability to connect and serve code from different domains that can connect seamlessly with partner subsystems. This cross domain connection is required to keep connections and callbacks registered into communication systems without any cross domain issues.

OpenFrame has two significant parts: one that lives in the ServiceNow application (referred to as TopFrame) and this API that is sourced from the partner application. This API has the necessary methods to communicate with TopFrame and control the visual features of the OpenFrame.

openFrameAPI - hide()

Hides the OpenFrame in the TopFrame.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

openFrameAPI.hide()

openFrameAPI - init(Object config, function successCallback, function failureCallback)

Initialize OpenFrame, must be the first method called.

This method initializes communication to TopFrame and initializes any visual elements passed in the config parameter.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config</td>
<td>Object</td>
<td>An object of key value pairs. The possible keys are height, width, title, subTitle, and titleIcon. All keys are optional.</td>
</tr>
<tr>
<td>successCallback</td>
<td>function</td>
<td>The callback function used if the init method succeeds. The openframe configuration stored in the system is passed as a parameter to the callback function.</td>
</tr>
<tr>
<td>failureCallback</td>
<td>function</td>
<td>The callback function used if the init method fails.</td>
</tr>
</tbody>
</table>
var config = {
  height: 300,
  width: 200
};

function handleCommunicationEvent(context) {
  console.log("Communication from Topframe", context);
}

function initSuccess(snConfig) {
  console.log("openframe configuration", snConfig);
  //register for communication event from TopFrame
  openFrameAPI.subscribe(openFrameAPI.EVENTS.COMMUNICATION_EVENT, handleCommunicationEvent);
}

function initFailure(error) {
  console.log("OpenFrame init failed..", error);
}

openFrameAPI.init(config, initSuccess, initFailure);

openFrameAPI - isVisible(function callback)
Checks to see if the OpenFrame is visible in the TopFrame.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>function</td>
<td>The callback function receives a parameter with a value of true or false. True if OpenFrame is visible and false if not visible.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

function callback(isVisible) {
  console.log(isVisible)
}
openFrameAPI.isVisible(callback)

openFrameAPI - openCustomURL(String details)
Opens a custom URL in TopFrame.

function openFrameAPI.openCustomURL(details) {
  // Implementation...
}
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Url</td>
<td>String</td>
<td>A string of 2083 or fewer characters.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
openFrameAPI.openCustomURL('10_cool_things.do');
```

**openFrameAPI - openServiceNowForm(Object details)**

Opens a form URL in TopFrame.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>details</td>
<td>Object</td>
<td>An object of key value pairs. The possible keys are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• entity, the table name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• query, an encoded query string</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
openFrameAPI.openServiceNowForm({entity:'case', query:'sys_id=1234567'});
```

**openFrameAPI - openServiceNowList(Object details)**

Opens a list URL in TopFrame.
openFrameAPI.openServiceNowList({entity: 'case', query: 'active=true'});

**openFrameAPI - setIcons(Array icons)**

The OpenFrame header can include icons that are placed next to the close icon.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>icons</td>
<td>Array</td>
<td>A list of icon configurations, where each icon configuration is an object with key values imageUrl, imageTitle, and any other needed context.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>


**openFrameAPI - setSize(Number width, Number height)**

Sets the OpenFrame size.
openFrameAPI - setSize(300, 370);

openFrameAPI - setSubtitle(String subTitle)
Sets the OpenFrame subtitle.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subTitle</td>
<td>String</td>
<td>A string of 256 or fewer characters.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

openFrameAPI.setSubtitle('+18888888888');

openFrameAPI - setTitle(String title)
Sets the OpenFrame title.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td>A string of 256 or fewer characters.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
openFrameAPI.setTitle('Incoming Call');
```

**openFrameAPI - setTitleIcon(Object icon)**

Sets the OpenFrame's title icon.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>icon</td>
<td>Object</td>
<td>Object of key value pairs. Keys include imageURL, imageTitle, and any other context needed.</td>
</tr>
</tbody>
</table>

```
openFrameAPI.setTitleIcon({imageURL:'/my/image/path.png',
                          imageTitle:'mute', id:101});
```

```
openFrameAPI.setTitleIcon({imageURL:'https://mydomain.com/image/path.png',
                          imageTitle:'mute', id:101});
```

**openFrameAPI - show()**

Makes the OpenFrame visible in the TopFrame.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
openFrameAPI - show()

openFrameAPI - subscribe(openFrameAPIEVENT event, function eventCallback)

Subscribes to the an event.

These events are available.

- openframe_header_icon_clicked -- this event occurs when any icon other than the close icon is clicked on the OpenFrame header. The callback receives the icon object as a parameter.
- openframe_shown -- this event occurs when the OpenFrame is shown.
- openframe_hidden -- this event occurs when the OpenFrame is hidden.
- openframe_before_destroy -- this event occurs before the TopFrame is unloaded.
- openframe_communication -- this event is application specific and can be customized.
- openframe_communication_failure -- this event occurs when communication to TopFrame fails.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>openFrameAPIEVENT</td>
<td>One of the available events.</td>
</tr>
<tr>
<td>eventCallback</td>
<td>function</td>
<td>The method called when the specified event occurs.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

function handleHeaderIconClick(context) {
  console.log("Header Icon was clicked", context);
}
openFrameAPI.subscribe(openFrameAPI.EVENTS.HEADER_ICON_CLICKED, handleHeaderIconClick);

openFrameAPI - version()

Returns the OpenFrame API version.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The OpenFrame API version</td>
</tr>
</tbody>
</table>

```javascript
var version = openFrameAPI.version();
console.log("API version "+ version);
```

**OAuthUtil**

This script include modifies request parameters and parses the token response during runtime. If the external OAuth provider returns a response other than an application/JSON type response, you can customize your own version of this script include to parse responses that are in different format. Extend or copy this script include, and then reference your version from the OAuth API Script field on the Application Registry form for third-party OAuth providers. The custom script include name must start with `OAuth`. For example, if the OAuth provider requires a resource parameter with value `https://outlook.office365.com`, the code would look like:

```javascript
preprocessAccessToken: function(requestParamMap) {
  requestParamMap.put("resource", "https://outlook.office365.com");
},
```

**OAuthUtil - interceptRequestParameters(requestParamMap)**

Adds a name:value pair to the request parameters.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestParamMap</td>
<td>String</td>
<td>The name:value pair you want to add.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**OAuthUtil - parseTokenResponse(accessTokenResponse)**

Parses the token received into a parameter map.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accessTokenResponse</td>
<td>String</td>
<td>The access token response that you want to parse.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**OCRosterSpanApprovalUtil**

The OCRosterSpanApprovalUtil API for the PTO approval feature.

**OCRosterSpanApprovalUtil - approvePTOSpan (GlideRecord rosterSpanGr)**

Changes the type of the roster_schedule_span to time-off from approval.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rosterSpanGr</td>
<td>GlideRecord</td>
<td>A roster_schedule_span_proposal record.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**OCRosterSpanApprovalUtil - getContextualCalUrlPerSpanProposal (GlideRecord rosterSpanProposalGr)**

Returns a URL to the on-call calendar based on the rosterSpanProposalGr.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rosterSpanProposalGr</td>
<td>GlideRecord</td>
<td>A roster_schedule_span_proposal record.</td>
</tr>
<tr>
<td>Returns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>A URL to the on-call calendar based on the rosterSpanProposalGr.</td>
<td></td>
</tr>
</tbody>
</table>

**OCRosterSpanApprovalUtil - getFromDateDisplayValuePerSpanProposal (GlideRecord rosterSpanProposalGr)**

Returns a formatted GlideScheduleDateTime string based on the start date of the rosterSpanProposalGr.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>rosterSpanProposalGr</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>String</td>
</tr>
</tbody>
</table>

**OCRosterSpanApprovalUtil - getPTOApproversList (GlideRecord rosterSpanProposalGr)**

Returns an array of group managers based on the user’s rota groups.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>rosterSpanProposalGr</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Array</td>
</tr>
</tbody>
</table>

**OCRosterSpanApprovalUtil - getToDateDisplayValuePerSpanProposal (rosterSpanProposalGr GlideRecord)**

Returns a formatted GlideScheduleDateTime string based on the end date of the rosterSpanProposalGr.
OCRosterSpanApprovalUtil - getUserNamePerSpanProposal (GlideRecord rosterSpanProposalGr)

Returns the user's name based on rosterSpanProposalGr.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rosterSpanProposalGr</td>
<td>GlideRecord</td>
<td>A roster_schedule_span_proposal record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A formatted GlideScheduleDateTime string based on the end date of the rosterSpanProposalGr.</td>
</tr>
</tbody>
</table>

OCRosterSpanApprovalUtil - isPTOApprovalRequired ()

Checks whether the system property com.snc.on_call_rotation.pto.approval.required is true.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the system property com.snc.on_call_rotation.pto.approval.required is true.</td>
</tr>
</tbody>
</table>
**OCRosterSpanApprovalUtil - rejectPTOSpan (rosterSpanGr GlideRecord)**

Changes the type of the roster_schedule_span to rejected from approval.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rosterSpanGr</td>
<td>GlideRecord</td>
<td>A roster_schedule_span_proposal record.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**OCRotation**

The core **OCRotation** builds a data structure that is used to display the calendar.

The code builds up the GlideAJAXSchedulePage object which stores a list of span items. You can use this type for the Gwt (legacy) calendar. See the OCRotationV2 API for other calendar types. This is part of On-Call Scheduling.

**OCRotation - buildRotas()**

Builds the on-call coverage based on your groups, rotas, and rosters. This is used by the OCRotationV2 - getSpans method.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void</td>
<td></td>
</tr>
</tbody>
</table>

**OCRotation - getEndDate()**

Returns the end date for the time period for which you want to retrieve on-call coverage.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>End date for the time period for which you want to retrieve on-call coverage.</td>
<td></td>
</tr>
</tbody>
</table>

### OCRotation - getGroupIds()

Returns a comma separated list of group sys_id (sys_user_group) values for an on-call schedule.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Comma separated list of group sys_id (sys_user_group) values for an on-call schedule.</td>
</tr>
</tbody>
</table>

### OCRotation - getRosterIds()

Returns the rosters according to the selected rotas.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Rosters according to the selected rotas.</td>
</tr>
</tbody>
</table>

### OCRotation - getRotaGr(String rotaIds, String groupIds, String rosterIds, String userIds)

Returns a GlideRecord for the cmn_rota table filtered by groups, rota, users, and roster.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotaIds</td>
<td>String</td>
<td>Comma separated list of rota sys_id (cmn_rota) values for an on-call schedule.</td>
</tr>
<tr>
<td>groupIds</td>
<td>String</td>
<td>Comma separated list of group sys_id (sys_user_group) values for an on-call schedule.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>rosterIds</td>
<td>String</td>
<td>Comma separated list of roster sys_id (cmn_rota_roster) values for an on-call schedule.</td>
</tr>
<tr>
<td>userIds</td>
<td>String</td>
<td>Comma separated list of user sys_id (sys_user) values for an on-call schedule.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>A GlideRecord for the cmn_rota table filtered by groups, rota, users, and roster.</td>
</tr>
</tbody>
</table>

**OCRotation - getRotaIds()**

Returns the rotas for the groups on your calendar.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Rotas for the groups on your calendar.</td>
</tr>
</tbody>
</table>

**OCRotation - getStartDate()**

Returns the start date for the time period for which you want to retrieve on-call coverage.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Start date for the time period for which you want to retrieve on-call coverage.</td>
</tr>
</tbody>
</table>

**OCRotation - getTimezone()**

Returns the timezone that your on-call calendar will use.
| Parameters | | | |
|---|---|---|
| Name | Type | Description |
| None | | |

| Returns | | |
|---|---|
| Type | Description |
| String | Timezone that your on-call calendar will use. For example, Europe/London and US/Pacific. |

**OCRotation - getUserIDs()**

Filters the schedules return by users. Returns comma separated list of user sys_id (sys_user) values used to filter on-call schedules.

| Parameters | | | |
|---|---|---|
| Name | Type | Description |
| None | | |

| Returns | | |
|---|---|
| Type | Description |
| String | Comma separated list of user sys_id (sys_user) values used to filter on-call schedules. |

**OCRotation - setEndDate(String endDate, Boolean inclusive)**

Sets the end date for the time period for which you want to retrieve on-call coverage. If the end date is not set, then the last day of the next month is applied by default. Use in conjunction with OCRotationV2 - getSpans.

| Parameters | | | |
|---|---|---|
| Name | Type | Description |
| endDate | String | (Optional) End date of the on-call schedule. |
| inclusive | Boolean | (Optional) When true, the end date is included in the time span. If not specified, this value defaults to true. |

| Returns | | |
|---|---|
| Type | Description |
| Void | |
**OCRotation - setGroupIds(String groupIds)**
Filters the schedules return by groups. Sets comma separated list of group sys_id (sys_user_group) values for an on-call schedule. Use in conjunction with OCRotationV2 - getSpans.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupIds</td>
<td>String</td>
<td>Comma separated list of group sys_id (sys_user_group) values for an on-call schedule.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void</td>
<td></td>
</tr>
</tbody>
</table>

**OCRotation - setRosterIds(String rosterIds)**
Sets the rosters according to the selected rotas. Use in conjunction with OCRotationV2 - getSpans.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RosterIds</td>
<td>String</td>
<td>Rosters according to the selected rotas.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**OCRotation - setRotaIds(String rotaIds)**
Sets the rotas for the groups on your calendar. Use in conjunction with OCRotationV2 - getSpans.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RotaIds</td>
<td>String</td>
<td>Rotas for the groups on your calendar.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void</td>
<td></td>
</tr>
</tbody>
</table>
**OCRotation - setStartDate(String startDate)**

Sets the start date for the time period for which you want to retrieve on-call coverage. If the start date is not set, then the first day of the previous month is applied by default. Use in conjunction with OCRotationV2 - getSpans.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startDate</td>
<td>String</td>
<td>(Optional) Start date for the time period for which you want to retrieve on-call coverage.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**OCRotation - setTimezone(String timezone)**

Sets the timezone that your on-call calendar will use. Use in conjunction with OCRotationV2 - getSpans.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timezone</td>
<td>String</td>
<td>Timezone that your on-call calendar will use. For example, Europe/London and US/Pacific.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void</td>
<td></td>
</tr>
</tbody>
</table>

**OCRotation - setUserIds(String userIds)**

Sets comma separated list of user sys_id (sys_user) values used to filter on-call schedules. Use in conjunction with OCRotationV2 - getSpans.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserIds</td>
<td>String</td>
<td>Comma separated list of user sys_id (sys_user) values used to filter on-call schedules.</td>
</tr>
</tbody>
</table>
OCRotationV2

OCRotationV2 is an extension of OCRotation, which is used by Fullcalendar.io and the DHTMLX On-Call Calendar.

OCRotationV2 is used to get the schedule time span between two dates that are further filtered by groups, rotas, rosters, and users. You can also extend OCRotationV2 to add or modify behavior. This is part of On-Call Scheduling.

OCRotationV2 - getGroups()

Returns an array of groups that have active rotas.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Array | An array of group objects. Each contains the following attributes:  
· sys_id of the sys_user_group record  
· name of the sys_user_group record |

OCRotationV2 - getRostersByRotas(String rotaSysIds)

Returns an array of active rosters for a given rota ID.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotaSysIds</td>
<td>String</td>
<td>Comma separated list of rota sys_id (cmn_rota) values.</td>
</tr>
</tbody>
</table>
OCRotationV2 - getRotasByGroup(String groupSysIds)

Returns an array of active rotas for a given group ID.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupSysIds</td>
<td>String</td>
<td>Comma separated list of group sys_id (sys_user_group) values.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of rota objects. Each contains the following attributes:</td>
</tr>
<tr>
<td></td>
<td>• sys_id of the cmn_rota_roster record</td>
</tr>
<tr>
<td></td>
<td>• name of the cmn_rota_roster record</td>
</tr>
<tr>
<td></td>
<td>• rota sys_id of the cmn_rota record</td>
</tr>
</tbody>
</table>

OCRotationV2 - getSpans()

Get the spans from the specified start date to the specified end date. If no start and end dates are provided, the start date defaults to a month before and the end date defaults to a month after the current time. You can also use the groups IDs, rota IDs, roster IDs, user IDs to further filter the spans.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of spans. Each contains the following attributes: id, sys_id, table, rota_id, roster_id, user_id, text, description, color, textColor, start_date, end_date.</td>
</tr>
</tbody>
</table>

Get all spans for the default time period

```javascript
var spans = new OCRotationV2().getSpans();
var firstSpanStartDate = spans[0].start_date; // get the first span's start date
```

Get all spans between 1st April 2014 and 5th June 2014

```javascript
var spans = new OCRotationV2()
  .setStartDate("2014-04-01")
  .setEndDate("2014-06-05")
  .getSpans();
```

Get the Network group's spans for the default time period

```javascript
var spans = new OCRotationV2()
  .setGroupIds("287ebd7da9fe198100f92cc8d1d2154e")
  .getSpans();
```

Get ITIL User's spans between 1st January 2014 and 31st January 2014

```javascript
var spans = new OCRotationV2()
  .setStartDate("2014-01-01")
  .setEndDate("2014-01-31")
  .setUserIds("681b365ec0a80164000fb0b05854a0cd")
  .getSpans();
```

**OCRotaICalendarSNC**

The OCRotaICalendarSNC API provides iCal formatted events for a specific user's rotation.

**OCRotaICalendarSNC - cleanExpiredCache ()**

Records in the cmn_rota_resp_cache table that have a From date prior to today.
OCRotaICalendarSNC - createCalendarEvents (String groupId, String rotaId, String userId, Object dateRangeObj)

Returns list of AJAXScheduleItems for the user’s on-call.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>String</td>
<td>The group sys id.</td>
</tr>
<tr>
<td>rotaId</td>
<td>String</td>
<td>The rota sys id.</td>
</tr>
<tr>
<td>userId</td>
<td>String</td>
<td>The user sys id.</td>
</tr>
<tr>
<td>dateRangeObj</td>
<td>Object</td>
<td>Contains the from and to dates for the data set.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJAXScheduleItem</td>
<td>Java list of AJAXScheduleItems.</td>
</tr>
</tbody>
</table>

OCRotaICalendarSNC - createCustomEvent (String groupName, Object scheduleItemSpan, String calendarLink)

Creates a VEVENT based on the scheduleItemSpan provided.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupName</td>
<td>String</td>
<td>Name of the on-call group.</td>
</tr>
<tr>
<td>scheduleItemSpan</td>
<td>Object</td>
<td>Span of time for the on-call period.</td>
</tr>
<tr>
<td>calendarLink</td>
<td>String</td>
<td>A link back to the on-call calendar in the instance.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A VEVENT based on the scheduleItemSpan provided.</td>
</tr>
</tbody>
</table>

**OCRotaICalendarSNC - createPlaceholderCalendar (GlideRecord rotaGR, Object dateRangeObj, String calendarLink)**

Creates a formatted iCalendar if the user has no upcoming on-call.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotaGR</td>
<td>GlideRecord</td>
<td>The cmn_rota glide record.</td>
</tr>
<tr>
<td>dateRangeObj</td>
<td>Object</td>
<td>Contains the from and to dates for the data set.</td>
</tr>
<tr>
<td>calendarLink</td>
<td>String</td>
<td>A link to the on-call calendar for this user.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Formatted iCalendar with a placeholder VEVENT.</td>
</tr>
</tbody>
</table>

**OCRotaICalendarSNC - getCalendarEvents (String groupId , String rotaId, String userId , Object dateRangeObj, Boolean useCache)**

Returns the formatted iCalendar for the user's rotation.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>String</td>
<td>The group sys id.</td>
</tr>
<tr>
<td>rotaId</td>
<td>String</td>
<td>The rota sys id.</td>
</tr>
<tr>
<td>userId</td>
<td>String</td>
<td>The user sys id.</td>
</tr>
<tr>
<td>dateRangeObj</td>
<td>Object</td>
<td>Contains the from and to dates for the data set.</td>
</tr>
<tr>
<td>useCache</td>
<td>Boolean</td>
<td>If true, makes use of the cache table cmn_rota_resp_cache.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Formatted iCalendar for the user's rotation.</td>
</tr>
</tbody>
</table>
**OCRotaICalendarSNC - getEventsFromTable (String groupId, String rotaId, String userId, Object dateRangeObj)**

Returns the formatted iCalendar for the user's rotation if found in the cmn_rota_resp_cache table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>String</td>
<td>The group sys id.</td>
</tr>
<tr>
<td>rotaId</td>
<td>String</td>
<td>The rota sys id.</td>
</tr>
<tr>
<td>userId</td>
<td>String</td>
<td>The user sys id.</td>
</tr>
<tr>
<td>dateRangeObj</td>
<td>Object</td>
<td>Contains the from and to dates for the data set.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Formatted iCalendar for the user's rotation if found in the cmn_rota_resp_cache table.</td>
</tr>
</tbody>
</table>

**OCRotaICalendarSNC - getIntersectRotaSpanItem (ScheduleTimeSpan timeSpan, Array rotaSpanItems)**

Checks whether the provided timeSpan intersects with one of the rota span items.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeSpan</td>
<td>ScheduleTimeSpan</td>
<td>A single span for the user's on-call.</td>
</tr>
<tr>
<td>rotaSpanItems</td>
<td>Array</td>
<td>Contains the user's standard on-call spans.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>If the timeSpan intersects with one of the rota span items, returns an object containing the rota span item and the span that it intersects with.</td>
</tr>
</tbody>
</table>

**OCRotaICalendarSNC - getMemberCalendarURL (String groupId, Object rotaId, String userId)**

Creates the subscribable URL for the user's iCalendar.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>String</td>
<td>The group sys id.</td>
</tr>
<tr>
<td>rotaId</td>
<td>Object</td>
<td>The rota sys id.</td>
</tr>
<tr>
<td>userId</td>
<td>String</td>
<td>The user sys id.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The subscribable URL for the user’s iCalendar.</td>
</tr>
</tbody>
</table>

**OCRotaICalendarSNC - getOnCallCalendarURL (GlideRecord rotaGR)**

Returns a URL to the on-call calendar based on the rota record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotaGR</td>
<td>GlideRecord</td>
<td>A cmn_rota record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>URL to the on-call calendar based on the rota record.</td>
</tr>
</tbody>
</table>

**OCRotaICalendarSNC - handleOverrideMember (AJAXScheduleItem scheduleItem)**

Returns an array of the user’s overrides (extra coverage).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleItem</td>
<td>AJAXScheduleItem</td>
<td>A list of AJAXScheduleItems.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of the user’s overrides (extra coverage).</td>
</tr>
</tbody>
</table>

**OCRotaICalendarSNC - handleRotaMember (AjaxScheduleItem scheduleItem , Object rotaSpanItems, Object definitionItems, Array repeatRotaSpanIdArr)**

Returns an object containing the user’s override (extra coverage) and excluded spans.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleItem</td>
<td>AJAXScheduelItem</td>
<td>A list of AJAXScheduelItems.</td>
</tr>
<tr>
<td>rotaSpanItems</td>
<td>Object</td>
<td>Contains the user's standard on-call spans.</td>
</tr>
<tr>
<td>definitionItems</td>
<td>Object</td>
<td>Contains the rota's standard on-call spans.</td>
</tr>
<tr>
<td>repeatRotaSpanIdArr</td>
<td>Array</td>
<td>An array of rota span sys ids.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Contains the user's override (extra coverage) and excluded spans.</td>
</tr>
</tbody>
</table>

### OCRotaICalendarSNC - invalRotaRespCache (GlideRecord rotaRespCacheGR)

Used by business rules on the cmn_rota_member, cmn_schedule_span, and roster_schedule_span tables to update the cmn_rota_resp_cache table.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotaRespCacheGR</td>
<td>GlideRecord</td>
<td>Record that has been updated.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### OCRotaICalendarSNC - matchRotaSpanRule(ScheduleTimeSpan timeSpan, Array rotaSpanItems, Array repeatRotaSpanIdArr, Object seriesStartTimes)

Checks whether the provided timeSpan matches one of the rotaSpanItems.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeSpan</td>
<td>ScheduleTimeSpan</td>
<td>A single span for the user's on-call.</td>
</tr>
<tr>
<td>rotaSpanItems</td>
<td>Array</td>
<td>Contains the user's standard on-call spans.</td>
</tr>
<tr>
<td>repeatRotaSpanIdArr</td>
<td>Array</td>
<td>An array of rota span sys ids.</td>
</tr>
<tr>
<td>seriesStartTimes</td>
<td>Object</td>
<td>Start time of the rota spans.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the timeSpan matches one of the rotaSpanItems.</td>
</tr>
</tbody>
</table>

**OCRotaICalendarSNC - populateCalendarSubscriptionSettings()**

Used by fixed job to populate the calendar subscription fields on the cmn_rota form

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**OCRotaICalendarSNC - processSeriesEvent(List scheduleItems, Array repeatRotaSpanIdArr, String userId)**

Returns an object containing the user's rotations.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleItems</td>
<td>List</td>
<td>The user's on-call rotation as an AJAXScheduleItem list.</td>
</tr>
<tr>
<td>repeatRotaSpanIdArr</td>
<td>Array</td>
<td>An array of the cmn_rota schedule span record sys_ids.</td>
</tr>
<tr>
<td>userId</td>
<td>String</td>
<td>The user sys id.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Contains the user's rotations.</td>
</tr>
</tbody>
</table>

**OCRotaICalendarSNC - saveCalendarEvents(String groupId, String rotaId, String userId, Object dateRangeObj, String result)**

Inserts the user's iCalendar into the cmn_rota_resp_cache table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>String</td>
<td>The group sys id.</td>
</tr>
</tbody>
</table>
### rotaId
- **Type**: String
- **Description**: The rota sys id.

### userId
- **Type**: String
- **Description**: The user sys id.

### dateRangeObj
- **Type**: Object
- **Description**: Contains the from and to dates for the data set.

### result
- **Type**: String
- **Description**: The user’s iCalendar.

### Returns
- **Type**: void

---

**OCRotaICalendarSNC - sendCalendarURL (GlideRecord rotaGR)**

Queues events to send an email to all members of the rotation.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotaGR</td>
<td>GlideRecord</td>
<td>A cmn_rota record.</td>
</tr>
</tbody>
</table>

#### Returns
- **Type**: void

---

**OCRotaICalendarSNC - updateExceptionList (AJAXScheduleItem scheduleItem, ScheduleTimeSpan timeSpan, Array rotaSpanItems)**

Returns an object that contains all of the timeSpans that need to be excluded from the user’s iCalendar.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleItem</td>
<td>AJAXScheduleItem</td>
<td>One AJAXScheduleItem.</td>
</tr>
<tr>
<td>timeSpan</td>
<td>ScheduleTimeSpan</td>
<td>A single user’s standard on-call span.</td>
</tr>
<tr>
<td>rotaSpanItems</td>
<td>Array</td>
<td>Contains the user’s standard on-call spans.</td>
</tr>
</tbody>
</table>
Object

Object that contains all of the timeSpans that need to be excluded from the user's iCalendar.

OCRotaCalendar

The OCRotaCalendar API is an extension of the OCRotaCalendarSNC API.

OCRotaMember

The OCRotaMember API performs maintenance operations on the cmn_rota_member table.

OCRotaMember - deactivateUser (String userId, GlideDateTime deactivateDate)

Deactivates the user's rota member records according to the deactivate date.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userId</td>
<td>String</td>
<td>The sys id of the user.</td>
</tr>
<tr>
<td>deactivateDate</td>
<td>GlideDateTime</td>
<td>The date from which the user's rotation is deactivated.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

OCRotaMember - hasChanged (GlideRecord current, GlideRecord previous)

Checks whether the cmn_rota_member record has changed. Checks the Member, From, To, and Order fields have changed.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current</td>
<td>GlideRecord</td>
<td>The latest changes made to the cmn_rota_member record.</td>
</tr>
<tr>
<td>previous</td>
<td>GlideRecord</td>
<td>The original state of the cmn_rota_member record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
OCRotaMember - hasOrderChanged (GlideRecord current, GlideRecord previous)
On update of a cmn_rota_member record, checks whether the order of the record has changed.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current</td>
<td>GlideRecord</td>
<td>The latest changes made to the cmn_rota_member record.</td>
</tr>
<tr>
<td>previous</td>
<td>GlideRecord</td>
<td>The original state of the cmn_rota_member record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the order has changed.</td>
</tr>
</tbody>
</table>

OCRotaMember - recalculate (GlideRecord current, GlideRecord previous)
Recalculates the schedule for the cmn_rota_member record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current</td>
<td>GlideRecord</td>
<td>The latest changes made to the cmn_rota_member record.</td>
</tr>
<tr>
<td>previous</td>
<td>GlideRecord</td>
<td>The original state of the cmn_rota_member record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

OCRotaMember - validateDates (String from, String to)
Ensures that the From date occurs before or on the same date as the To date.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>String</td>
<td>Formatted date/</td>
</tr>
<tr>
<td>to</td>
<td>String</td>
<td>Formatted date/</td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the From date is before or on same date as To date.</td>
</tr>
</tbody>
</table>

### OnCallRotation

Use to manage on-call schedules.

### OnCallRotation - getCatchAll(String rotaID)

Gets the sys_id of the catch all person. Use together with the `getCatchAllType()` method to determine the source of the returned id. Get the sys_id of the catch all persons.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotaID</td>
<td>String</td>
<td>The sys_id of the rota.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of a sys_user, a cmn_rota_roster, or null.</td>
</tr>
</tbody>
</table>

```javascript
var rotaID = "...";
var rota = new SNC.OnCallRotation()
gs.log("catch all: " + rota.catchAll(rotaID) + " (type:" +
        rota.catchAllType(rotaID) + ")");
```

### OnCallRotation - getCatchAllType(String rotaID)

Gets the type of _catch all_ that lives at the end of the escalation chain. When the escalation chain finishes without assigning an on-call person, if configured, the incident is assigned to the _catch all_.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotaID</td>
<td>String</td>
<td>The sys_id of the rota.</td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>- Null: invalid rota id or catch all is not available.</td>
</tr>
<tr>
<td></td>
<td>- Group_manager: the manager of the group linked to the rota is the catch all person.</td>
</tr>
<tr>
<td></td>
<td>- Individual: a configured user is the catch all person.</td>
</tr>
<tr>
<td></td>
<td>- All: all members of the configured roster are the catch all person.</td>
</tr>
</tbody>
</table>

```javascript
var rotaID = "...";
var rota = new SNC.OnCallRotation()
gs.log("catch all: " + rota.catchAll(rotaID) + " (type:" +
  rota.catchAllType(rotaID) + ")");
```

### OnCallRotation - getEscalateeAt(String groupId, GlideDateTime dateTime, Number position)

Gets an item to escalate to, either a user (sys_user) or notification device (cmn_notif_device), for a group's rota at the specified date and time and at a certain position in the escalation lineup.

The function returns null if an invalid position or group is passed as an argument.

You can use the `getTableName()` function to check whether the returning GlideRecord comes from the sys_user or cmn_notif_device table.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>String</td>
<td>The sys_id for the group from which to get the item to escalate.</td>
</tr>
<tr>
<td>dateTime</td>
<td>GlideDateTime</td>
<td>The date and time when the escalation lineup should begin.</td>
</tr>
<tr>
<td>position</td>
<td>Number</td>
<td>The position in the lineup to determine the item to escalate to (1-based, which means the count starts with 1).</td>
</tr>
</tbody>
</table>
OnCallRotation - getEscalationPlan(String groupID, GlideDateTime dateTime)

Gets a list of objects to escalate to, escalation plans that consist of either a user (sys_user) or notification device (cmn_notif_device), for a group's rota at the specified date and time.

The Escalation object contains the following fields:

- Number order: the escalation order within the lineup
- String userId: sys_id of the sys_user record or null
- String deviceId: sys_id of the cmn_notif_device record or null
- Boolean isDevice: true or false depending on whether the item to escalate to is a device or user
- GlideDuration timeBetweenReminders: the time between reminders being sent
- Number reminderNum: number of reminders to be sent for each item to escalate to

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupID</td>
<td>String</td>
<td>The sys_id for the group to get the to escalate to from.</td>
</tr>
<tr>
<td>dateTime</td>
<td>GlideDateTime</td>
<td>The date and time when the escalation lineup should begin.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>A list of escalation objects to escalate to.</td>
</tr>
</tbody>
</table>

var oncall = new SNC.OnCallRotation();
var escalatee =
oncall.getEscalateeAt("287ebd7da9fe198100f92cc8d1d2154e",
    new GlideDateTime("2013-11-26 09:00:00"), 2);

if (escalatee){
    gs.print(escalatee.getTableName());
    gs.print(escalatee.getTableName());
}

var rota = new SNC.OnCallRotation()
var gdt = new GlideDateTime()
gdt.addMonths(1);//Next month
```javascript
var escalationPlan =
rota.getEscalationPlan("486ae95aeb201100fcfb858ad106fe40", gdt);

for(var i = 0; i < escalationPlan.size(); i++){
gs.log(escalationPlan.get(i).order + " " +
escalationPlan.get(i).userId + " " +
escalationPlan.get(i).deviceId + " " +
escalationPlan.get(i).timeBetweenReminders + " " +
escalationPlan.get(i).reminderNum );
}
```

**OnCallRotation - getEscalationType(String rotaID)**

Gets the type of escalation based on a simple calculation. If there is one active roster in the rota, rotate through members. If there is more than one active roster in the rota, rotate through rosters.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotaID</td>
<td>String</td>
<td>The sys_id for the rota to check for rosters.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>• Member: if there is one roster in the rota.</td>
</tr>
<tr>
<td></td>
<td>• Roster: if there is more than one roster in the rota.</td>
</tr>
<tr>
<td></td>
<td>• No rosters in this rota: if none of the above.</td>
</tr>
</tbody>
</table>

```javascript
var rota = new SNC.OnCallRotation() //contains one roster
gs.log(rota.getEscalationType("486ae95aeb201100fcfb858ad106fe40")); // logs 'member'
```

**OnCallRotation - OnCallRotation()**

Creates an instance of the OnCallRotation class.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var oncall = new SNC.OnCallRotation();
```
OnCallRotation - spansOverlap(ArrayList parent, ArrayList child, String timeZone)

Checks if the parent span and child spans overlap.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>ArrayList</td>
<td>The parent spans to compare.</td>
</tr>
<tr>
<td>child</td>
<td>ArrayList</td>
<td>The child spans to compare.</td>
</tr>
<tr>
<td>timeZone</td>
<td>String</td>
<td>The name of the time zone.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the parent and child spans overlap.</td>
</tr>
</tbody>
</table>

```javascript
var onCall = new SNC.OnCallRotation();

// get a time range we are interested in
var timeZone = gs.getSession().getTimeZoneName();
var dateStart = new GlideDateTime();
var dateEnd = new GlideDateTime().addMonths(1);

// convert glidedatetime to scheduledatet ime because it is required
// by the function being called below
var scheduleStart = new GlideScheduleDateTime(dateStart);
var scheduleEnd = new GlideScheduleDateTime(dateEnd);
scheduleStart.setTimeZone(timeZone);
scheduleEnd.setTimeZone(timeZone);

// calculate rotation items based on the date and time given
// for all the groups the currently logged in user is a member of
gs.include("OnCallRotationPersonal");
var rotation = new OnCallRotationPersonal();
var myGroups = gs.getUser().getMyGroups();
var groupIter = myGroups.iterator();

while(groupIter.hasNext()){
    var rotaItems = rotation.onCallDuringPeriod(groupIter.next(),
        scheduleStart, scheduleEnd);

    // loop through rotation schedules
    for (var i = 0; i < rotaItems.length - 1; i++) {
        var parentItem = rotaItems[i];
        var nextItem = rotaItems[i + 1];

        // check if previous and next time spans overlap
        var overlaps = onCall.spansOverlap(parentItem.getTimeSpans(),
            nextItem.getTimeSpans(),
            gs.getSession().getTimeZoneName());
    }
}
**OCSeriesEventGenerator**

The OCSeriesEventGenerator API generates the repeating VEVENTS for a user’s iCalendar.

**OCSeriesEventGenerator - getMemberCalendar**

```
(String groupId, String rotaId, String userId, Object dateRangeObj, Object memberSchedules, String calendarLink)
```

Returns an array of the repeating on-call events.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>String</td>
<td>The group sys id.</td>
</tr>
<tr>
<td>rotaId</td>
<td>String</td>
<td>The rota sys id.</td>
</tr>
<tr>
<td>userId</td>
<td>String</td>
<td>The user sys id.</td>
</tr>
<tr>
<td>dateRangeObj</td>
<td>Object</td>
<td>Contains the from and to dates for the data set.</td>
</tr>
<tr>
<td>memberSchedules</td>
<td>Object</td>
<td>The user’s schedule.</td>
</tr>
<tr>
<td>calendarLink</td>
<td>String</td>
<td>A link back to the on-call calendar in the instance.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>Array of the repeating on-call events.</td>
</tr>
</tbody>
</table>

**OCSeriesEventGenerator - getRosterICalEvents**

```
(GlideRecord rotaScheduleSpanGR, GlideRecord memberScheduleSpanGR, Array seriesStartTimes, Object excludeItems, String repeatUntil, String timeZone, String calendarLink)
```

Returns an array of VEVENTs that represent the on-call for the provided rotaScheduleSpanGR and the memberScheduleSpanGR.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotaScheduleSpanGR</td>
<td>GlideRecord</td>
<td>A cmn_schedule_span glide record for a rota.</td>
</tr>
<tr>
<td>memberScheduleSpanGR</td>
<td>GlideRecord</td>
<td>A cmn_schedule_span glide record for a rota member.</td>
</tr>
<tr>
<td>seriesStartTimes</td>
<td>Array</td>
<td>An array of the repeating rota span start times.</td>
</tr>
<tr>
<td>excludeItems</td>
<td>Object</td>
<td>The schedule items that need to be excluded.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repeatUntil</td>
<td>String</td>
<td>A formatted ScheduleDateTime string.</td>
</tr>
<tr>
<td>timeZone</td>
<td>String</td>
<td>Time zone for the on-call rota.</td>
</tr>
<tr>
<td>calendarLink</td>
<td>String</td>
<td>A link to the on-call calendar for this user.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of VEVENTs that represent the on-call for the provided rotaScheduleSpanGR and the memberScheduleSpanGR.</td>
</tr>
</tbody>
</table>

**OCSeriesEventGenerator - getRosterMemberEvents**

(GlideRecord rotaMemberGR, Array seriesStartTimes, Object excludeItems, String repeatUntil, String calendarLink)

Returns an array of the repeating on-call VEVENTS.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotaMemberGR</td>
<td>GlideRecord</td>
<td>A cmn_rota_member record.</td>
</tr>
<tr>
<td>seriesStartTimes</td>
<td>Array</td>
<td>An array of the repeating rota span start times.</td>
</tr>
<tr>
<td>excludeItems</td>
<td>Object</td>
<td>The schedule items that need to be excluded.</td>
</tr>
<tr>
<td>repeatUntil</td>
<td>String</td>
<td>A formatted ScheduleDateTime string.</td>
</tr>
<tr>
<td>calendarLink</td>
<td>String</td>
<td>A link to the on-call calendar for this user.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of the repeating on-call VEVENTS</td>
</tr>
</tbody>
</table>

**OCTimer**

The OCTimer API provides the ability to track the length of time it takes for a function to complete.

**OCTimer - log**

(String name, String entry)

Logs the entry based on the function name.
### OCTimer - millisToTime (Integer millis)

Takes a milisecond value and returns a formatted duration.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>millis</td>
<td>Integer</td>
<td>Duration in milliseconds.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A formatted duration.</td>
</tr>
</tbody>
</table>

### OCTimer - result ()

Calculates the time taken for each function registered.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Time taken for each function registered.</td>
</tr>
</tbody>
</table>
### OCTimer - stop (String name)

Registers the end of the timer for the provided name.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Function name.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### OrderGuide - Scoped

OrderGuide API enables you to initialize and view an order guide details. To use this class in a scoped application, use the sn_sc namespace identifier. The Service Catalog Scoped API plugin (com.glideapp.servicecatalogscoped.api) that is enabled by default is required to access the OrderGuide API.

**OrderGuide - OrderGuide(String sys_id)**

Creates an instance of the OrderGuide class with the specified sys_id.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>sys_id of the OrderGuide.</td>
</tr>
</tbody>
</table>

**Example:**

```javascript
var cart=new sn_sc.OrderGuide("6690750f4f7b4200086eeed18110c761");
```

**OrderGuide - getID()**

Returns the sys_id of the order guide.
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>sys_id of the order guide.</td>
</tr>
</tbody>
</table>

### Example:

```
var cart=new sn_sc.OrderGuide("6690750f4f7b4200086eed18110c761");
console.log(cart.getID());
```

Output:

```
6690750f4f7b4200086eed18110c761
```

## OrderGuide - init(Map request)

Initialises the order guide with the specified catalog items and the variables, and returns the order guide.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>Map</td>
<td>A JSON object with the Catalog item and variable details.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>A JSON object with the initialised order guide details.</td>
</tr>
</tbody>
</table>

### Example:

```
var guide = new sn_sc.OrderGuide('6690750f4f7b4200086eed18110c761');
var map = {};
map.variables = {};
//map.sysparm_id = '6690750f4f7b4200086eed18110c761';
map.variables['IOce433d0f4f7b4200086eed18110c74d'] = '221f3db5c6112284009f4becd3039cc9'; //Here ce433d0f4f7b4200086eed18110c74d is the sys_id of the variable and 221f3db5c6112284009f4becd3039cc9 is its value
```
var includedItems = guide.init(map)
{
  "cascade_variable_map":{
  },
  "items":[
  {
    "order": "-1",
    "quantity": "1",
    "show_quantity": false,
    "sys_id": "a4022d7b87c20300e3010cf888cb0bb2"
  },
  {
    "order": "300",
    "quantity": "1",
    "show_quantity": false,
    "sys_id": "186d917a6fab7980575967d8bb3ee4f2"
  },
  {
    "order": "600",
    "quantity": "1",
    "show_quantity": false,
    "sys_id": "8b3ae7fedc1be1004ece5c08239e522b"
  }
  ],
  "variable_assignments":{
  }
}

OrderGuide - isIncludeItems()

Specifies if the Show Include Toggle (include_items) check box is selected for the specified order guide.

Note: The Show Include Toggle field does not appear on the Order guide form by default.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the Show Include Toggle check box is selected for the specified order guide. Else, returns false.</td>
</tr>
</tbody>
</table>

Example

```javascript
var orderGuide = new sn_sc.OrderGuide("6690750f4f7b420086eed18110c761");
console.log(orderGuide.isIncludeItems());
```
Output:

```
true
```

**OrderGuide - isTwoStep()**

Specifies if the two-step checkout is enabled.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the two-step checkout is enabled. Else returns false.</td>
</tr>
</tbody>
</table>

**Example:**

```javascript
var orderGuide=new sn_sc.OrderGuide("6690750f4f7b4200086eeed18110c761");
console.log(orderGuide.isTwoStep());
```

Output:

```
false
```

**OrderGuide - isUseCustomCart()**

Specifies if a separate cart (different from that for catalog items) usage is enabled for a two-step order guide.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if a separate cart usage is enabled for a two-step order guide. Else, returns false.</td>
</tr>
</tbody>
</table>
Example:

```javascript
var orderGuide=new
sn_sc.OrderGuide("6690750f4f7b4200086eeed18110c761");
console.log (orderGuide.isUseCustomCart());
```

Output:

```javascript
false
```

**OrderGuide - navigateFromMap(Map itemDetails)**

Navigates to the catalog items of an order guide.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemDetails</td>
<td>Map</td>
<td>A JSON object with details of catalog items in the order guide.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example:

```javascript
var orderGuide=new
sn_sc.OrderGuide.navigateFromMap(itemdetails);
```

**PADomainUtils**

The PADomainUtils API enables you to copy Performance Analytics configurations between different domains.

Use this API in server scripts to copy Performance Analytics configuration records, such as indicators, breakdowns, and dashboards, to different domains. This API enables you to create a Performance Analytics configuration in one domain and copy that configuration to any number of additional domains.

**Note:** This API cannot copy records into the Global domain.

To use PADomainUtils, you must satisfy these requirements:

- Performance Analytics must be licensed.
- The user running the script must have the admin role.
- The instance must use domain separation.
- The script must be run from the global domain.
- When moving or copying records, the source and target domains must be different.
PADomainUtils - copy(String runAs)

Copies Performance Analytics configuration records to a different domain.
To copy dashboards or scheduled jobs, see copyDashboard and copyJob.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>runAs</td>
<td>String</td>
<td>The user whose domain you want to copy records to.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
// copy all the Performance Analytics records from global to user's domain
var pa = new SNC.PADomainUtils();
pa.copy('09ff3d105f231000b12e3572f2b4775d');
```

PADomainUtils - copyDashboard(String dashboardId, String runAs)

Copy a dashboard to another domain.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashboardId</td>
<td>String</td>
<td>The sys_id of the dashboard to copy.</td>
</tr>
<tr>
<td>runAs</td>
<td>String</td>
<td>The user whose domain you want to copy the dashboard to.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Optional example explanation

```java
//Copy Incident Management dashboard from global to user's domain
var pa = new SNC.PADomainUtils();
pa.copyDashboard('a64b7031d7201100b96d45a3ce610335','09ff3d105f231000b12e3572f2b4775d');
```
**PADomainUtils - copyJob(String paJob, String runAs)**

Copies a Performance Analytics scheduled data collection job record to another domain.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paJob</td>
<td>String</td>
<td>The sys_id of a Performance Analytics scheduled data collection job [sysauto_pa] record.</td>
</tr>
<tr>
<td>runAs</td>
<td>String</td>
<td>The user whose domain you want to copy the job to.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>An error message if an error occurs, or an empty string if there is no error.</td>
</tr>
</tbody>
</table>

```java
// No source domain needs to be set
var pa = new SNC.PADomainUtils();
// copy the OOTB '[PA Incident] Daily Data Collection job'
// set the 'run as' of the new record to be the 'acme.itil' user
// first argument is the sys_id of the sysauto_pa record
// the second is the sys_id of the acme.itil user record
pa.copyJob('82ba2023d7101100b96d45a3ce6103cd','797d14341f1310005a3637b8ec8b7010');
```

**PADomainUtils - isWriteable(String table, String id)**

Evaluate if you can write to a specific record identified by table and sys_id.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>The name of the table containing the record to query, such as pa_indicators.</td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td>The sys_id of the record to query.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Indicates that you can write to the specified record. Returns true if the record exists within the domain of the current user. Returns false if the record does not exist, or is in a different domain.</td>
</tr>
</tbody>
</table>

```javascript
var pa = new SNC.PADomainUtils();
pa.isWriteable('pa_incidents','cd8125b5140012007665a83e633b028d');
```

**PADomainUtils - move(String runAs)**

Moves Performance Analytics configuration records to a different domain.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>runAs</td>
<td>String</td>
<td>The user whose domain you want to copy records to.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
// move all the Performance Analytics records from the global to the customers domain
var pa = new SNC.PADomainUtils();
pa.move('774190f01f1310005a3637b8ec8b70ef');
```

**PADomainUtils - PADomainUtils()**

Instantiates a new PADomainUtils object to move or copy Performance Analytics configuration records from the global domain.

Use the PADomainUtils(String domainFrom) constructor instead when moving or copying records from a domain other than the global domain.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
// PADomainUtils initialized with the global domain
```
var globalUtils = new SNC.PADomainUtils();

### PADomainUtils - PADomainUtils(String domainFrom)

Instantiates a new PADomainUtils object to move or copy Performance Analytics configuration records from the specified domain.

Use the PADomainUtils() constructor instead when moving or copying from the global domain.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domainFrom</td>
<td>String</td>
<td>The domain to copy records from.</td>
</tr>
</tbody>
</table>

// c90d4b084a362312013398f051272c0d is the sys id of the ACME domain
var acmeUtils = new SNC.PADomainUtils('c90d4b084a362312013398f051272c0d');

### PADomainUtils - setFoundation(Boolean foundation)

Uses this method to move or copy only foundation records in a hybrid domain configuration.

You can implement a hybrid configuration by maintaining some types of record in a parent domain and some types in child domains. Records maintained in the parent domain are known as foundation records. The following types of record are considered foundation records.

- Bucket groups
- Buckets
- Scripts
- Breakdown sources
- Indicator sources
- Filters
- Breakdowns
- Managed sources
- Manual breakdowns
- Breakdown mappings
- Breakdown relations

Other Performance Analytics configuration records such as widgets and indicators are not foundation records. Set this method to false to move or copy these additional records as well.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>foundation</td>
<td>Boolean</td>
<td>Indicates if only foundation records should be copied or moved by this PADomainUtils object.</td>
</tr>
</tbody>
</table>
### PADomainUtils - setOverrides(Boolean overrides)

Use this method before copying records to set the sys_override value of the new record to the original parent record.

Using this method enables you to automatically override records in a parent domain. By overriding the parent records, the parent records do not impact the child domain. If the source domain is not the parent of the target domain when copying records, setting the sys_override value will not have any impact on behavior. You can specify an override only when copying records, not when moving records.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>overrides</td>
<td>Boolean</td>
<td>Indicates that copied records in a child domain should override the source record in the parent domain. This value is true by default.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PADomainUtils</td>
<td>The object calling this function.</td>
</tr>
</tbody>
</table>

```javascript
var pa = new SNC.PADomainUtils().setFoundation(true);
pa.copy('bb6b58b01f1310005a3637b8ec8b70dd');
```

```javascript
var pa = new SNC.PADomainUtils('c90d4b084a362312013398f051272c0d');
pa.setOverrides(false);
pa.copy('bb6b58b01f1310005a3637b8ec8b70dd');
```

### PAScorecard

The PAScorecard API enables you to query information about Performance Analytics scorecards and indicators.

#### PAScorecard - addParam(String parameter, String value)

Add a query parameter to filter the returned scores.
Call this method multiple times on the same PAScorecard object to pass multiple parameters, such as the indicator sys_id and a breakdown sys_id. After specifying all parameters, call query() to run the query and get the resulting scorecard object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>String</td>
<td>The parameter to set. For a detailed list of available parameters, see PAScorecard parameters.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to assign to the specified parameter.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var sc = new SNC.PAScorecard(); // in a scoped app, do not use the SNC namespace
sc.addParam('uuid', 'fb007202d7130100b96d45a3ce6103b4'); // Number of open incidents
sc.addParam('breakdown', '0df47e02d7130100b96d45a3ce610399'); // by Priority
var result = sc.query();
var json = sc.asJSON();
for (var i = 0; i < result.length; i++)
gs.info(result[i].name + ': ' + result[i].value + ' ' + result[i].unit.display_value);
```

**PAScorecard - asJSON()**

Returns the latest query result as a JSON string.

This method does not perform a query. To perform a query before returning the result, use query().

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON representation of the query result.</td>
</tr>
</tbody>
</table>
**PAScorecard - query()**

Performs a query based on the specified parameters and return the scorecard as an object.

Before calling this method, configure parameters for the PAScorecard object by calling `addParam(String parameter, String value)`.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Object</td>
</tr>
</tbody>
</table>

**PAScorecard - result()**

Returns the latest query result as an object.

This method does not perform a query. To perform a query before returning the result, use `query()`.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Object</td>
</tr>
</tbody>
</table>

**PASnapshot**

The PASnapshot API enables you to query information about Performance Analytics snapshots.

You can query information about a snapshot at a certain date using the indicator `sys_id` and `date`, and perform comparisons between snapshots for an indicator at different dates.

**PASnapshot - getCompareIDs(String sys_id, Number date1, Number date2, String type)**

Compare records in snapshots for a specified indicator at multiple dates, such as to identify records included in one snapshot but not the other.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>sys_id</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>The indicator sys_id.</td>
</tr>
<tr>
<td>date1</td>
<td>Number</td>
<td>The date of the first snapshot, in the format yyyymmdd.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A comma-separated list of sys_id values.</td>
</tr>
</tbody>
</table>

```javascript
var snapshot2 = PASnapshot.getCompareIDs('fb007202d7130100b96d45a3ce6103b4', 20160430, 20160531, 'shared');
gs.info(snapshot2);
```

Output: *** Script: 09c01200d7002100b81145a3ce6103ab,19c01200d7002100b81145a3ce6103e9,fcc01200d7002100b81145a3...

### PASnapshot - getCompareQuery(String sys_id, Number date1, Number date2, String type)

Get the query used to compare records in snapshots for a specified indicator at multiple dates.
ServiceNow

Kingston

Now Platform Custom Business Applications

Name

Type

Description

date2

Number

The date of the second
snapshot, in the format
yyyymmdd.

type

String

Specifies what data to retrieve.
Valid values are:
• all1: all records in the first
snapshot
• all2: all records in the second
snapshot
• shared: records that are in
both snapshots
• movedin: records that are in
the second snapshot, but not
the first
• movedout: records that are
in the first snapshot, but not
the second

Returns

Type

Description

String

The table, view, and encoded query as a JSON
string.

var snapshot4 =
PASnapshot.getCompareQuery('fb007202d7130100b96d45a3ce6103b4',
20160530, 20160531, 'all1');
gs.info(snapshot4);
Output: *** Script: {"view":"","query":"sys_idINjavascript:new
PAUtils().getCompareSnapshotIDs(\"fb007202d7130100b96d45a3ce6103b4\",
\"20160530\",\"20160531\",\"all1\")","table":"incident"}

PASnapshot - getIDs(String sys_id, Number date)
Get the sys_id values for all records contained in the snapshot for a specified indicator at the
specified date.
Parameters

Name

Type

Description

sys_id

String

The indicator sys_id.

date

Number

The date when the snapshot
was taken, in the format
yyyymmdd.

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States
and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.

1363


Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A comma-separated list of sys_id values.</td>
</tr>
</tbody>
</table>

```javascript
var snapshot1 = PASnapshot.getIDs('fb007202d7130100b96d45a3ce6103b4', 20160530);
gs.info(snapshot1);
```

Output: *** Script:
09c01200d7002100b81145a3ce6103ab,19c01200d7002100b81145a3ce6103e9,fcc01200d7002100b81145a3
....

**PASnapshot - getQuery(String sys_id, Number date)**

Get the query used to generate the snapshot for a specified indicator at the specified date.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>The indicator sys_id.</td>
</tr>
<tr>
<td>date</td>
<td>Number</td>
<td>The date when the snapshot was taken, in the format yyyymmd.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The table, view, and encoded query as a JSON string.</td>
</tr>
</tbody>
</table>

```javascript
var snapshot3 = PASnapshot.getQuery('fb007202d7130100b96d45a3ce6103b4', 20160530);
gs.info(snapshot3);
```

Output: *** Script: {
"view":"
"query":"sys_idINjavascript:new PAUtils().getSnapshotIDs("fb007202d7130100b96d45a3ce6103b4", "20160530")","table":"incident"}

**ProbeHandlerCim**

Configures a CIM probe.

Use in any server-side script where you need to configure a CIM probe.
**ProbeHandlerCim - getCimQueries()**

Returns a string containing the CIM fields to fetch.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The CIM fields to fetch</td>
</tr>
</tbody>
</table>

**ProbeHandlerCim - getNamespace()**

Returns the CIM name space.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The name space</td>
</tr>
</tbody>
</table>

**ProbeHandlerCim - getParameters()**

Returns a hash map of the parameters added by this probe handler.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A hash map of the parameters</td>
</tr>
</tbody>
</table>

**ProbeHandlerCim - setProbeParameters(Object params)**

Sets the probe parameters.
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>params</td>
<td>Object</td>
<td>The parameters to add</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerCim - run()**

Runs the probe.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerMulti**

Probe handler for MultiProbes.

This script includes does not work with probe parameters that have valueScripts and does not work with JavaScript probes.

Use in any server-side script where you need to configure a MultiProbe.

**ProbeHandlerMulti - addParameters(String parent, String sysID, GlideRecord probe)**

Adds the probe parameters to the given XML parent element, taken from the given probe's parameter table.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>String</td>
<td>The XML parent element to add parameters to.</td>
</tr>
<tr>
<td>sysID</td>
<td>String</td>
<td>The sys_id of the probe to get parameters from.</td>
</tr>
<tr>
<td>probe</td>
<td>GlideRecord</td>
<td>Optional GlideRecord of the probe. If present, and there is a probe handler, it is called.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerMulti - addProbes(String parent, String sysID)**

Adds the probes for this multiprobe to the document.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>String</td>
<td>The XML parent element to add the probes to.</td>
</tr>
<tr>
<td>sysID</td>
<td>String</td>
<td>The sys_id of the multi-probe</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerMulti - run()**

Runs the probe.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerMulti - runProbeHandler(String params, String sysID, GlideRecord probe)**

If the given probe’s GlideRecord is present and contains a probe handler, runs the probe handler and adds any parameters it creates to the given parameter element.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>params</td>
<td>String</td>
<td>The XML parameter element to add parameters to.</td>
</tr>
<tr>
<td>sysID</td>
<td>String</td>
<td>The sys_id of the probe to get parameters from.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>probe</td>
<td>GlideRecord</td>
<td>Optional GlideRecord of the probe. If present, and there is a probe handler, it is called.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### ProbeHandlerMulti - setParameter(String parent, String name, String value)

Sets the value of a given XML parameter element.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>String</td>
<td>The XML parent element to add the parameter to.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the parameter to set</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to set the parameter to.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

#### ProbeHandlerSNMP

Discovery probe handler for SNMP probes, invoked when an SNMP probe is about to be created. Use in any server-side script where you need to configure an SNMP probe.

**ProbeHandlerSNMP - addBoolean(String dbName, String paramName, String paramMap)**

Adds a boolean value to the parameter map.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbName</td>
<td>String</td>
<td>The database name</td>
</tr>
<tr>
<td>paramName</td>
<td>String</td>
<td>The parameter name</td>
</tr>
<tr>
<td>paramMap</td>
<td>String</td>
<td>The parameter map</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerSNMP - addDebug(String paramMap)**

Adds debugging to the probe handler.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramMap</td>
<td>String</td>
<td>The parameter map</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerSNMP - addJavascriptHandlers(String paramMap)**

Adds a JavaScript handler.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramMap</td>
<td>String</td>
<td>The parameter map</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerSNMP - addOIDAutoResolve(String paramMap)**

Adds OID automatic resolution.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramMap</td>
<td>String</td>
<td>The parameter map</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
**ProbeHandlerSNMP - addOIDSpec(String paramMap)**

Adds OID specifications to the parameter map.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramMap</td>
<td>String</td>
<td>The parameter map</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerSNMP - addSensorHandler(String paramMap)**

Adds a sensor handler.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramMap</td>
<td>String</td>
<td>The parameter map</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerSNMP - addStage(String paramMap)**

Adds the stage to the parameter map.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramMap</td>
<td>String</td>
<td>The parameter map</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerSNMP - addTimingSpecs(String paramMap)**

Adds timing specifications to the parameter map.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>paramMap</td>
<td>String</td>
<td>The parameter map</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerSNMP - getParameters()**

Returns the parameters added by this probe handler.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td></td>
</tr>
<tr>
<td>A hash map of the parameters added by this probe handler.</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerSNMP - run()**

Runs the probe.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerSNMP - setProbeParameters(String params)**

Sets the value of a given XML parameter element.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>params</td>
<td>String</td>
<td>The probe’s parameters</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerWMI**

Discovery probe handler for WMI probes.
Use in any server-side script where you need to configure an WMI probe.

**ProbeHandlerWMI - getParameters()**

Returns the parameters added by this probe handler.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A hash map of the parameters added by this probe</td>
</tr>
</tbody>
</table>

**ProbeHandlerWMI - getWMIFields()**

Generates a string containing the WMI fields to fetch.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The WMI fields to fetch.</td>
</tr>
</tbody>
</table>

**ProbeHandlerWMI - run()**

Runs the probe.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ProbeHandlerWMI - setProbeParameters(Object params)**

Sets the probe parameters.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>params</td>
<td>Object</td>
<td>The probe parameters</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Queue**

The Queue API allows you to retrieve or join a Connect Support chat queue.

To use this class in a scoped application, use the sn_connect namespace identifier. The Connect Scriptable APIs plugin (ID: com.glide.connect.scriptable) should be enabled to access the Queue API.

**Scoped Queue - get(String sysID)**

Get an existing chat queue by sys_ID.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysID</td>
<td>String</td>
<td>The sysID of a queue from the chat_queue table.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Returns a conversation queue object.</td>
</tr>
</tbody>
</table>

**Example:**

```javascript
var queue = sn_connect.Queue.get("ab73be7dc09a4300964f336ee6b74361");
```
**Scoped queue - join(String question)**

Adds the current user to an existing Connect Support chat queue. Use a sysID from the chat_queue table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question</td>
<td>String</td>
<td>Type a question to add to the queue.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example:

```javascript
var queue = sn_connect.Queue.get("ab73be7dc09a4300964f336ee6b74361");
queue.join("How do I access my email?");
```

**QuickRanges**

Generates IP network, range, and address entries from a convenient comma-separated input field using conventional CIDR network notation, hyphenated range entries, or individual IP addresses.

Use with any server-side discovery script.

**QuickRanges - createItem(String table, String id, String type)**

Creates a new discovery range item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>The table where the item will be created.</td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td>The identifier to use for the new item.</td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>The type of entries to generate: IP address, IP network, or IP range.</td>
</tr>
</tbody>
</table>
### GlideRecord

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>The created entry</td>
</tr>
</tbody>
</table>

### QuickRanges - onMakeRanges()

Returns the IP network, range, and address information to use when generating the entries.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### RecordToHTML

A utility class to turn a record into HTML.

The RecordToHTML class is available to server-side scripts.

#### RecordToHTML - RecordToHTML(String table, String sys_id, String pattern, Boolean link)

Creates an instance of RecordToHTML class.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>String</td>
<td>The record’s table name</td>
</tr>
<tr>
<td>sys_id</td>
<td>String</td>
<td>The sys_id of the record</td>
</tr>
<tr>
<td>pattern</td>
<td>String</td>
<td>The pattern of the string to generate. The pattern may include ${} escapes for fields whose values should be included. For example, the pattern ‘sys_id: ${sys_id}’ would substitute the actual sys_id for the escape.</td>
</tr>
<tr>
<td>link</td>
<td>Boolean</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var r2html = new RecordToHTML("incident","e8e875b0c0a80164009dc852b4d677d5", "incident: ${number}-${short_description}", true);
```
gs.print(r2html.toString());

Output: incident: INC00005-CPU load high for over 10 minutes

**RecordToHTML - setValue(String fieldName, String value)**

Sets the specified field to the specified value.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldName</td>
<td>String</td>
<td>Name of the field to change.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>Value to set the field to.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
var r2html = new RecordToHTML("incident","e8e875b0c0a80164009dc852b4d677d5", "incident: ${number}-${short_description} (${user})", true);
r2html.setValue("user",gs.getUserName());
gs.print(r2html.toString())
```

Output: incident: INC00005-CPU load high for over 10 minutes (john.roberts)

**RecordToHTML - toString()**

Converts the record to a string.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>HTML output of the record.</td>
</tr>
</tbody>
</table>

```
var r2html = new RecordToHTML("incident","e8e875b0c0a80164009dc852b4d677d5", "incident: ${number}-${short_description}", true);
```
gs.print(r2html.toString());

Output: incident: INC00005-CPU load high for over 10 minutes

**RenderProperties**

The RenderProperties API provides methods about the current page and is available in Jelly scripts and in UI-action conditions and scripts.

Access `RenderProperties` methods using the static variable `RP`.

**RenderProperties - getEncodedQuery()**

Returns the encoded query from the URL sent to the page.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Returns the encoded query from the URL sent to the form.</td>
</tr>
</tbody>
</table>

**RenderProperties - getListControl()**

Returns the list control object for the page.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysListControl object</td>
<td>The list control object for the page.</td>
</tr>
</tbody>
</table>

**RenderProperties - getParameterValue(String parameterName)**

Returns the value of the specified URL parameter.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameterName</td>
<td>String</td>
<td>Name of the parameter passed on the URL.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The parameter’s value.</td>
</tr>
</tbody>
</table>

**RenderProperties - getReferringURL()**

Returns the URL where the request originated.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The URL of the page where the request originated.</td>
</tr>
</tbody>
</table>

**RenderProperties - getViewID()**

Returns the view the page is using.

This method is not available in scoped applications.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the view being used.</td>
</tr>
</tbody>
</table>

**RenderProperties - getViewName()**

Returns the name of the view in use.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### RenderProperties - getWindowProperties()

Returns the window's properties.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The window's properties</td>
</tr>
</tbody>
</table>

### RenderProperties - isInDevStudio()

Returns true if the page is part of Studio.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the page is part of Studio.</td>
</tr>
</tbody>
</table>

### RenderProperties - isInteractive()

Returns true if this is an interactive session. An interactive session is when a user has logged in as opposed to a REST request.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if this is an interactive session.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
RenderProperties - isManyToMany()

Returns true when the `sysparm_collection_related_file` URL parameter is set.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true when the <code>sysparm_collection_related_file</code> URL parameter is set.</td>
</tr>
</tbody>
</table>

RenderProperties - isRelatedList()

Returns true when the `sys_is_related_list` URL-parameter is true. Returns false if the parameter is not present.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the URL parameter <code>sys_is_related_list</code> is true.</td>
</tr>
</tbody>
</table>

RenderProperties

The RenderProperties API provides methods about the current page and is available in Jelly scripts and in UI-action conditions and scripts.

Access `RenderProperties` methods using the static variable `RP`.

Scoped ScopedRenderProperties - getEncodedQuery()

Returns the encoded query from the URL sent to the page.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>Returns the encoded query from the URL sent to the form.</td>
<td></td>
</tr>
</tbody>
</table>

**Scoped ScopedRenderProperties - getListControl()**

Returns the list control object for the page.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Scoped ScopedRenderProperties - getParameterValue(String parameterName)**

Returns the value of the specified URL parameter.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameterName</td>
<td>String</td>
<td>Name of the parameter passed on the URL.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The parameter's value.</td>
</tr>
</tbody>
</table>

**Scoped ScopedRenderProperties - getReferringURL()**

Returns the URL where the request originated.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The URL of the page where the request originated.</td>
</tr>
</tbody>
</table>

**Scoped ScopedRenderProperties - getViewName()**

Returns the name of the view in use.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The name of the view being used.</td>
</tr>
</tbody>
</table>

**Scoped ScopedRenderProperties - getWindowProperties()**

Returns the window's properties.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The window's properties</td>
</tr>
</tbody>
</table>

**Scoped ScopedRenderProperties - isInDevStudio()**

Returns true if the page is part of Studio.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the page is part of Studio.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
### Scoped ScopedRenderProperties - isInteractive()

Returns true if this is an interactive session. An interactive session is when a user has logged in as opposed to a REST request.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if this is an interactive session.</td>
</tr>
</tbody>
</table>

### Scoped ScopedRenderProperties - isManyToMany()

Returns true when the `sysparm_collection_related_file` URL parameter is set.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true when the <code>sysparm_collection_related_file</code> URL parameter is set.</td>
</tr>
</tbody>
</table>

### Scoped ScopedRenderProperties - isRelatedList()

Returns true when the `sys_is_related_list` URL-parameter is true. Returns false if the parameter is not present.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the URL parameter <code>sys_is_related_list</code> is true.</td>
</tr>
</tbody>
</table>
RESTAPIRequest

A RESTAPIRequest object allows you to access scripted REST API request details in scripts.

![Note: You cannot instantiate objects of this type. Objects of this type are created automatically and are accessible only in scripted REST API resource scripts.]

RESTAPIRequest - body

The body of the request.

Field

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>body</td>
<td>RESTAPIRequestBody</td>
<td>The body of the request. You can access data from the body object using the RESTAPIRequestBody API.</td>
</tr>
</tbody>
</table>

```
var requestBody = request.body // Returns instance of RESTAPIRequestBody
```

RESTAPIRequest - getHeader(String header)

Returns the value of a specific header from the web service request.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>String</td>
<td>The name of the header, such as accept or content-type.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the specified header.</td>
</tr>
</tbody>
</table>

```
var acceptHeader = request.getHeader('accept');
```

RESTAPIRequest - getSupportedResponseContentTypes()

Get the content types specified in the request Accept header.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of string values where each string is a content type, such as application/json.</td>
</tr>
</tbody>
</table>

### RESTAPIRequest - headers

All headers from the request.

#### Field

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headers</td>
<td>object</td>
<td>All headers from the request, and their values.</td>
</tr>
</tbody>
</table>

```javascript
var headers = request.headers;
var acceptHeader = headers.Accept;
var myCustomHeader = headers.myCustom;
var specialHeader = headers['special - header'];
```

### RESTAPIRequest - pathParams

The path parameters passed in the request URI.

#### Field

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pathParams</td>
<td>Object</td>
<td>The path parameters as a script object. Available path parameters depend on the web service configuration.</td>
</tr>
</tbody>
</table>

In this example, the scripted REST API endpoint follows this format: `https://instance.service-now.com/api/now/myservice/{tableName}/{id}`. The request being processed uses this URL: `https://instance.service-now.com/api/now/myservice/myApp_table/1234`.

```javascript
var pathParams = request.pathParams;
var tableName = pathParams.tableName; // 'myApp_table'
var id = pathParams.id; // '1234'
```
RESTAPIRequest - queryParams
The query parameters from the web service request.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryParams</td>
<td>Object</td>
<td>The query parameters from the web service request.</td>
</tr>
</tbody>
</table>

In this example, the request being processed uses this URL: https://<instance_rest_endpoint>?active=false&name=now. Note the active and name parameters.

```javascript
var queryParams = request.queryParams;
var isActiveQuery = queryParams.active; //false
var nameQueryVal = queryParams.name; // 'now'
```

RESTAPIRequest - queryString
The entire query added to the endpoint URI.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryString</td>
<td>String</td>
<td>The entire query for the request.</td>
</tr>
</tbody>
</table>

In this example, the request being processed uses this URL: https://<instance_rest_endpoint>?active=false&name=now. Note the query active=false&name=now.

```javascript
var query = request.queryString; //"active=false&name=now"
```

RESTAPIRequest - uri
The request URI, excluding domain information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>String</td>
<td>The request URI, excluding domain information.</td>
</tr>
</tbody>
</table>
In this example, the request being processed uses this URL: https://instance.service-now.com/api/now/table/myTable?
active=false&name=now.

```javascript
var query = request.uri; //"api/now/table/myTable"
```

**RESTAPIRequest - url**

The entire request URL.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>String</td>
<td>The entire request URL.</td>
</tr>
</tbody>
</table>

In this example, the request being processed uses this URL: https://instance.service-now.com/api/now/table/myTable?
active=false&name=now.

```javascript
var query = request.url; //"https://instance.service-now.com/api/now/table/myTable?active=false&name=now"
```

**RESTAPIRequestBody**

A RESTAPIRequestBody object allows you to access the body content of a scripted REST API request in scripts.

The format of a RESTAPIRequestBody object may be JSON or XML, depending on the content-type header value from the request.

**Note:** You cannot instantiate objects of this type. Objects of this type are created automatically and are accessible only in scripted REST API resource scripts.

Single entry example-request-body in JSON format.

```json
{
    "name": "user1",
    "id": 1234,
    "roles": [
        { "name": "admin" },
        { "name": "itil" }
    ]
}
```

Multiple entry example-request-body in JSON format.

```json
[{
    "name": "user1",
    "id": 1234,
}]
```
"roles": [
  {
    "name": "admin"
  },
  {
    "name": "itil"
  },
  {
    "name": "user2",
    "id": 9876,
    "roles": [
      {
        "name": "admin"
      }
    ]
  }
],

Important: If the request body format is not Application/json, Application/xml, or Text/xml, use only the request body `dataStream` field to access the request body. Using request body `data`, `dataString`, `nextEntry()`, or `hasNext()` with a non-default format will result in a 500 error response.

RESTAPIRequestBody - data
The content of the request body.

Note: REST web services consume data, datastream, and dataString via stream. Because streams are only consumed once, only one can be called. For example, if calling both data and dataString, the second call will return empty.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>data</td>
<td>Object or Array</td>
<td>The request content. This can be a single object or an array of objects depending on the request.</td>
</tr>
</tbody>
</table>

```javascript
var entry;
var id;
var requestBody = request.body;
var requestData = requestBody.data; // May be an array or a single object
if (requestData instanceof Array) {
  entry = requestData[0].name; // 'user1'
  id = requestData[0].id; // '1234'
} else {
  entry = requestData.name; // 'user1'
  id = requestData.id; // '1234'
}
```
RESTAPIRequestBody - dataStream

The content of the request body, as a stream.

**Note:** REST web services consume data, datastream, and dataString via stream. Because streams are only consumed once, only one can be called. For example, if calling both data and dataString, the second call will return empty.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dataStream</td>
<td>Object</td>
<td>The content of the request body. You can pass the stream to a separate API, such as to create an attachment from the request or forward the request to a different endpoint.</td>
</tr>
</tbody>
</table>

```javascript
var requestBody = request.body;
var requestStream = requestBody.dataStream;
```

RESTAPIRequestBody - dataString

The content of the request body, as a String.

**Note:** REST web services consume data, datastream, and dataString via stream. Because streams are only consumed once, only one can be called. For example, if calling both data and dataString, the second call will return empty.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dataString</td>
<td>String</td>
<td>The content of the request body.</td>
</tr>
</tbody>
</table>

```javascript
var requestBody = request.body;
var requestString = requestBody.dataString;
```

RESTAPIRequestBody - hasNext()

Determine if there are additional entries in the request body.

Use this method with the nextEntry() method to iterate over multiple request body entries.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>True if there are additional entries available. This method returns true only once if the request contains a single entry.</td>
</tr>
</tbody>
</table>

var requestBody = request.body;
requestBody.hasNext(); // returns true if the request contains a single entry or multiple entries

// calling second time
requestBody.hasNext(); // returns false if the request contains a single entry, or true if the request contains multiple entries

**RESTAPIRequestBody - nextEntry()**

Retrieve one entry from the request body as a script object.

Use this method with the hasNext() method to iterate over multiple request body entries.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>A single entry from the request body.</td>
</tr>
</tbody>
</table>

var requestBody = request.body;
var requestEntry = requestBody.nextEntry(); // returns available entry if there is only one entry, or the first entry if there are multiple.
var name = requestEntry.name; // ‘user1’

// Calling second time
requestEntry = requestBody.nextEntry(); // returns undefined if there is only one entry, or the second entry if there are multiple.
This example demonstrates using hasNext() with nextEntry().

```javascript
var requestBody = request.body;
while(requestBody.hasNext()) {
    var entry = requestBody.nextEntry();
}
```

RESTAPIResponse

A RESTAPIResponse object allows you to build a RESTful response to a scripted REST API request.

**Note:** You cannot instantiate objects of this type. Objects of this type are created automatically and are accessible only in scripted REST API resource scripts.

RESTAPIResponse - getStreamWriter()

Get the ResponseStreamWriter for this response, allowing you to write directly to the response stream.

Set the content type and status code using the `setHeaders` and `setStatus` functions prior to calling the `getStreamWriter` function.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#unique_2601</td>
<td>The ResponseStreamWriter for this response. You can use this object to write directly to the response stream.</td>
</tr>
</tbody>
</table>

```javascript
response.setContentType('application/json');
response.setStatus(200);
var writer = response.getStreamWriter();
```

RESTAPIResponse - setBody(Object body)

Sets the body content to send in the web service response.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>body</td>
<td>Object</td>
<td>The response body, as a JavaScript object. The body content is automatically serialized to JSON or XML depending on the value of the Accept header passed in the request.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
var body = {};
body.name = "incident";
body.number = "1234";
body.caller = {"id": "user1"};
response.setBody(body);

var bodyArray = [];
var body = {};
body.name = "incident";
body.number = "1234";
body.caller = {"id": "user1"};
bodyArray.push(body);
response.setBody(bodyArray);
```

`RESTAPIResponse - setContentType(String contentType)`

Assigns a value to the Content-Type header in the web service response.

You must set a response content type before writing the response. The content type is set automatically for string responses, based on the request Accept header value.

Setting an invalid content type causes the response to default to JSON. Failing to set a content type results in a status code 500 error when sending a binary response.

See the [W3 Content-Type header documentation](#) for more information about this header.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contentType</td>
<td>String</td>
<td>The content type of the response body, such as application/json.</td>
</tr>
</tbody>
</table>
# ServiceNow Kingston Now Platform Custom Business Applications

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
responseBuilder.setContentType('application/json');
```

### RESTAPIResponse - setError(Object error)

Configure the response to return an error.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>Object</td>
<td>An error object.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### RESTAPIResponse - setHeader(String header, String value)

Assign a value to a REST service response header.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>String</td>
<td>The header you want to set.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to assign the specified header.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
responseBuilder.setHeader("Location","<URI>");
```

### RESTAPIResponse - setHeaders(Object headers)

Sets the headers for the web service response.

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headers</td>
<td>Object</td>
<td>A JavaScript object listing each header and the value to assign that header.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var headers = {};
headers.X-Total-Count=100;
headers.Location='https://instance.service-now.com/<endpoint_to_resource>';
response.setHeaders(headers);
```

### RESTAPIResponse - setLocation(String location)

Assigns a value to the Location header in the web service response.

See the [W3 Location header documentation](https://www.w3.org/TR/REXX-HTTP/headers/#location) for more information about this header.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>String</td>
<td>An absolute URI to redirect the response recipient to.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### RESTAPIResponse - setStatus(Number status)

Sets the status code number for the web service response.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Number</td>
<td>The status code to send in the response, such as 200 to indicate success. Passing a non-numerical value, such as a string, causes the status code to default to 0.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
response.setStatus(200);
```

### RESTAPIResponseStream

A RESTAPIResponseStream object allows you to write directly to the scripted REST API response stream.

Use RESTAPIResponseStream methods to build web service APIs in the Scripted REST APIs feature.

**Note:** You cannot instantiate objects of this type. Objects of this type are created automatically and are accessible only in scripted REST API resource scripts.

### RESTAPIResponseStream - writeStream(Object stream)

Write an input stream to the response stream.

You must set the content type and status code before calling the `writeStream()` method or the response will fail. You cannot modify these values after calling the `writeStream()` method.

**Note:**

It is the responsibility of the script author to obtain the stream from a third-party service.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream</td>
<td>Object</td>
<td>An attachment or a response stream from a third-party service.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

The following example is for scoped applications:

```javascript
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    response.setContentType('application/json');
    response.setStatus(200);

    var gsa = new GlideSysAttachment();
    var attachmentStream = new gsa.getContentStream(<sys_id of attachment>);
    var writer = response.getStreamWriter();
    writer.writeStream(attachmentStream);

})(request, response);
```

The following example is for global applications:

```javascript
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    response.setContentType('application/json');
    response.setStatus(200);

    var attachmentStream = new GlideSysAttachmentInputStream(<sys_id of attachment>);
    var writer = response.getStreamWriter();
    writer.writeStream(attachmentStream);

})(request, response);
```

**RESTAPIResponseStream - writeString(String data)**

Write string data to the response stream.

You must set the content type and status code before calling the `writeString()` method or the response will fail. You cannot modify these values after calling the `writeString()` method.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>String</td>
<td>The string to add to the response data.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
response.setContentType('application/json');
response.setStatus(200);
var writer = response.getWriter();
var body = {'
    name: user1,
    id: 1234,
    roles: [
        { name: admin },
        { name: itil }
    ]
'}
writer.writeString("{'name':'user','id':'1234'}");
writer.writeString(JSON.stringify(body));
```

RESTMessageV2

The RESTMessageV2 API allows you to send outbound REST messages using JavaScript. Use the RESTResponseV2 API to manage the response returned by the REST provider. You can use this API in scoped applications, or within the global scope.

RESTMessageV2 - execute()

Sends the REST message to the endpoint.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTResponse</td>
<td>The response returned by the REST provider.</td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```java
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); // Might throw exception if message doesn't exist or not visible due to scope.
```
var response = sm.execute(); // Might throw exception if HTTP connection timed out or some issue with sending request itself because of encryption/decryption of password.

**RESTMessageV2 - executeAsync()**

Sends the REST message to the endpoint asynchronously. The instance does not wait for a response from the web service provider when making asynchronous calls.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTResponse</td>
<td>The response returned by the REST provider.</td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>", "get"); // Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.executeAsync(); // Might throw exception if HTTP connection timed out or some issue with sending request itself because of encryption/decryption of password.
response.waitForResponse(60); // In seconds. Wait at most 60 seconds to get response from ECC Queue/Mid Server // Might throw exception timing out waiting for response in ECC queue.
```

**RESTMessageV2 - getEndpoint()**

Returns the URL of the endpoint for the REST message.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The URL of the REST web service provider.</td>
</tr>
</tbody>
</table>
In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); //Might throw exception if message doesn't exist or not visible due to scope.
var endpoint = sm.getEndpoint();
```

**RESTMessageV2 - getRequestBody()**

Returns the content of the REST message body.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The REST message body.</td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); //Might throw exception if message doesn't exist or not visible due to scope.
var body = sm.getRequestBody();
```

**RESTMessageV2 - getRequestHeader(String headerName)**

Returns the value for an HTTP header specified in the REST message.

By default, this method cannot return the value for a header set automatically by the system. To grant this method access to all headers, set the property `glide.http.log_debug` to true.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headerName</td>
<td>String</td>
<td>The request header you want to get the value for.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------------------</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>The value of the specified header.</td>
<td></td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); // Might throw exception if message doesn't exist or not visible due to scope.
var header = sm.getRequestHeader("Accept");
```

**RESTMessageV2 - getRequestHeaders()**

Returns HTTP headers that were set by the REST client and the associated values.

This method does not return headers set automatically by the system. To configure this method to return all headers, set the property `glide.http.log_debug` to true.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An Object that maps the name of each header to the associated value.</td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); // Might throw exception if message doesn't exist or not visible due to scope.
var headers = sm.getRequestHeaders();
```

**RESTMessageV2 - RESTMessageV2()**

Instantiates an empty RESTMessageV2 object.
When using an object instantiated this way, you must manually specify an HTTP method and endpoint.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.RESTMessageV2();
```

**RESTMessageV2 - RESTMessageV2(String name, String methodName)**

Instantiates a RESTMessageV2 object using information from a REST message record.

You must have a REST message record defined before you can use this constructor.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the REST message record.</td>
</tr>
<tr>
<td>methodName</td>
<td>String</td>
<td>Name of the HTTP method to use, such as GET/get or PUT/put - case-insensitive.</td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); // Might throw exception if message doesn't exist or not visible due to scope.
```

**RESTMessageV2 - saveResponseBodyAsAttachment(String tableName, String recordSysId, String fileName)**

Configures the REST message to save the returned response body as an attachment record.

When you use this function with a REST message that is sent through a MID server, the MID server user must have any roles required to read and write attachment records, as well as any roles required to read and write records on the table specified in the `tableName` parameter.

The response body does not need to be a binary file to be saved as an attachment. Response bodies using text formats, such as JSON or XML can also be saved. If the instance fails to save the attachment, call `getErrorMessage()` on the related RESTResponseV2 object for error details.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Specify the table that contains the record you want to attach the saved file to.</td>
</tr>
</tbody>
</table>
### Name | Type | Description
--- | --- | ---
recordSysId | String | Specify the sys_id of the record you want to attach the saved file to.
fileName | String | Specify the file name to give to the saved file.

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
(function sampleRESTMessageV2() {
  try{
    var request  = new sn_ws.RESTMessageV2();
    request.setHttpMethod('get');

    var attachment_sys_id = '<attachment_record_sys_id>',
    tablename = 'incident',
    recordSysId = '<incident_sys_id>',
    response,
    httpResponseStatus,
    filename = '<filename>';

    //endpoint - ServiceNow REST Attachment API
    request.setEndpoint('https://<instance_name>.service-now.com/api/now/attachment/' + attachment_sys_id + '/file');

    request.setBasicAuth('<username>', '<password>');

    //RESTMessageV2 - saveResponseBodyAsAttachment(String tableName, String recordSysId, String fileName)
    request.saveResponseBodyAsAttachment(tablename, recordSysId, filename);

    response = request.execute();
    httpResponseStatus = response.getStatusCode();

    gs.print(" http response status_code: " + httpStatus); 
  }
  catch(ex){
    var message = ex.getMessage();
    gs.print(message);
  }
})();
```

**RESTMessageV2 - saveResponseBodyAsAttachment(String tableName, String recordSysId, String fileName, String encryptContext)**

Configure the REST message to save the returned response body as an encrypted attachment record.

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
When you use this function with a REST message that is sent through a MID server, the MID server user must have any roles required to read and write attachment records, as well as any roles required to read and write records on the table specified in the `tableName` parameter.

The response body does not need to be a binary file to be saved as an attachment. Response bodies using text formats, such as JSON or XML can also be saved. If the instance fails to save the attachment, call `getErrorMessage()` on the related RESTResponseV2 object for error details.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Specify the table that contains the record you want to attach the saved file to.</td>
</tr>
<tr>
<td>recordSysId</td>
<td>String</td>
<td>Specify the sys_id of the record you want to attach the saved file to.</td>
</tr>
<tr>
<td>fileName</td>
<td>String</td>
<td>Specify the file name to give to the saved file.</td>
</tr>
<tr>
<td>encryptContext</td>
<td>String</td>
<td>Specify the sys_id of an encryption context. The saved file is encrypted using this context.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**RESTMessageV2 - setAuthenticationProfile(String type, String profileId)**

Sets the credentials for the REST message using an existing basic auth or OAuth 2.0 profile.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>The type of authentication profile to use. Valid values are 'basic' to use basic authentication, or 'oauth2' to use OAuth 2.0.</td>
</tr>
<tr>
<td>profileId</td>
<td>String</td>
<td>The sys_id of an authentication profile record. When using basic auth, specify the sys_id of a Basic Auth Configuration (sys_auth_profile_basic) record. When using OAuth 2.0, specify the sys_id of a OAuth Entity Profile (oauth_entity_profile) record.</td>
</tr>
</tbody>
</table>
In the following example, replace REST_message_record with the name of the REST message record from your instance.

```javascript
var requestBody;
var responseBody;
var status;
var sm;
try{
    // Might throw exception if message doesn't exist or not visible due to scope.
    sm = new sn_ws.RESTMessageV2("<REST_message_record>", "get");

    // set auth profile to an OAuth 2.0 profile record.
    sm.setAuthenticationProfile('oauth2', '1234adsf123212311123qasdfs');
    sm.setStringParameter("symbol", "NOW");
    sm.setStringParameterNoEscape("xml_data","<data>test</data>");

    // In milliseconds. Wait at most 10 seconds for response from http request.
    sm.setHttpTimeout(10000);
    // Might throw exception if http connection timed out or some issue
    // with sending request itself because of encryption/decryption of password.
    response = sm.execute();
    responseBody = response.haveError() ? response.getErrorMessage() : response.getBody();
    status = response.getStatusCode();
} catch(ex) {
    responseBody = ex.getMessage();
    status = '500';
} finally {
    requestBody = sm ? sm.getRequestBody():null;
}
```

**RESTMessageV2 - setBasicAuth(String userName, String userPass)**

Sets basic authentication headers for the REST message.

Setting security values using this method overrides basic authentication values defined for the REST message record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userName</td>
<td>String</td>
<td>The username you want to use to authenticate the REST message.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>userPass</td>
<td>String</td>
<td>The password for the specified user.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setBasicAuth("username","password");
```

**RESTMessageV2 - setEccCorrelator(String correlator)**

Associate outbound requests and the resulting response record in the ECC queue. This method only applies to REST messages sent through a MID Server.

The correlator provided populates the Agent correlator field on the ECC queue record for the response. Provide a unique correlator for each outbound request to associate the correct results in the ECC queue with the request when designing asynchronous automation through a MID Server.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>String</td>
<td>A unique identifier</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setEccCorrelator("unique_identifier");
```
RESTMessageV2 - setEccParameter(String name, String value)

Override a value from the database by writing to the REST message payload. This method only applies to REST messages sent through a MID Server.

Use this method when a value from the REST message in the database is invalid, such as when the endpoint URL is longer than the maximum REST endpoint field length. You can set only the endpoint URL using this method by passing source as the name parameter.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the parameter, such as source.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to assign to the specified parameter.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

In the following example, replace REST_message_record with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); //Might throw exception if message doesn't exist or not visible due to scope.
sm.setEccParameter("source","http://very.long.endpoint.url");
```

RESTMessageV2 - setEndpoint(String endpoint)

Set the endpoint for the REST message.

By default, the REST message uses the endpoint specified in the REST message record. Use this method to override this default. You must call this method when using the `RESTMessageV2` constructor with no parameters.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
<td>String</td>
<td>The URL of the REST provider you want to interface with.</td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.RESTMessageV2();
sm.setEndpoint("http://web.service.endpoint");
```

### RESTMessageV2 - setHttpMethod(String method)

The HTTP method this REST message performs, such as GET or PUT.

You must set an HTTP method when using the `RESTMessageV2` constructor with no parameters.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>String</td>
<td>The HTTP method to perform.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.RESTMessageV2();
sm.setHttpMethod("post");
```

### RESTMessageV2 - setHttpTimeout(Number timeoutMs)

Set the amount of time the REST message waits for a response from the web service provider before the request times out.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeoutMs</td>
<td>Number</td>
<td>The amount of time, in milliseconds, before the call to the REST provider times out.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); //Might throw exception if message doesn't exist or not visible due to scope.
sm.setHttpTimeout(6000);
```

**RESTMessageV2 - setLogLevel(String level)**

Set the log level for this message and the corresponding response.

Setting a log level using the RESTMessageV2 API overrides the log level configured on the REST message record. This log level may not apply if the endpoint domain is blacklisted, or if the property `glide.outbound_http_log.override` is true. To view outbound web service logs, navigate to System Logs > Outbound HTTP Requests.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>level</td>
<td>String</td>
<td>The log level. Valid values are basic, elevated, and all.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var rm = new sn_ws.RESTMessageV2();
rm.setLogLevel('all');
```

**RESTMessageV2 - setMIDServer(String midServer)**

Configures the REST message to communicate through a MID Server.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>midServer</td>
<td>String</td>
<td>The name of the MID Server to use. Your instance must have an active MID Server with the specified name.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setMIDServer("mid_server_name");
```

**RESTMessageV2 - setMutualAuth(String profileName)**

Sets the mutual authentication protocol profile for the REST message.

Setting a protocol profile using this method overrides the protocol profile selected for the REST message record.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>profileName</td>
<td>String</td>
<td>The Name of the protocol profile to use for mutual authentication.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setMutualAuth("mutual_auth_profile_name");
```

**RESTMessageV2 - setQueryParameter(String name, String value)**

Append a parameter to the end of the request URL with the form name=value.

For example, the code `setQueryParameter("sysparm_query","active=true\ORDERBYnumber\ORDERBYDESCcategory")` appends the text `sysparm_query=active=true^ORDERBYnumber^ORDERBYDESCcategory` to the request URL.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the URL parameter to pass.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to assign the URL parameter.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.RESTMessageV2();
//Set up message, including endpoint and authentication
sm.setQueryParameter("sysparm_query","active=true^ORDERBYnumber^ORDERBYDESCcategory");
```

RESTMessageV2 - setRequestBody(String body)

Set the body content to send to the web service provider when using PUT or POST HTTP methods.

When you set the body content using this method, variables in the body are not substituted for parameters from the REST message function record. You must explicitly define all values within the REST message body.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>body</td>
<td>String</td>
<td>The request body to send.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.RESTMessageV2("Update user","post"); //Might throw exception if message doesn't exist or not visible due to scope.
var body = "<Message body content>";
sm.setRequestBody(body);
```

RESTMessageV2 - setRequestBodyFromAttachment(String attachmentSysId)

Sets the request body using an existing attachment record.

When you use this function with a REST message that is sent through a MID server, the MID server user must have any roles required to read attachment records.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attachmentSysId</td>
<td>String</td>
<td>The sys_id of the Attachment (sys_attachment) record you want to send in this REST message.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
(function sampleRESTMessageV2() {
  try {
    var request = new sn_ws.RESTMessageV2();
    request.setHttpMethod('post');
    request.setEndpoint('<web service endpoint URL>');
    request.setRequestBodyFromAttachment('<attachment sys_id>');

    var response = request.execute();
    var httpResponseStatus = response.getStatusCode();

    gs.print("http response status_code: " +
              httpResponseStatus);
  }
  catch (ex) {
    var message = ex.getMessage();
    gs.print(message);
  }
})();
```

**RESTMessageV2 - setRequestHeader(String name, String value)**

Sets an HTTP header in the REST message to the specified value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the header.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to assign to the specified header.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
In the following example, replace \textit{REST\_message\_record} with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setRequestHeader("Accept","Application/json");
```

\textbf{RESTMessageV2 - setRequestorProfile(String requestorContext, String requestorId)}

Override the default requestor profile for the REST message in order to retrieve an OAuth access token associated with a different requestor.

This method applies only to REST messages configured to use OAuth 2.0 authentication. This method is optional and is unnecessary in most configurations.

\textbf{Parameters}

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestorContext</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>requestorId</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

\textbf{Returns}

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

\textbf{RESTMessageV2 - setStringParameter(String name, String value)}

Sets a REST message function variable with the specified name from the REST message record to the specified value.

XML reserved characters in the value are converted to the equivalent escaped characters. Use \texttt{setStringParameterNoEscape} to set a variable without escaping XML reserved characters.

\textbf{Parameters}

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the REST message variable. This parameter must be defined in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REST message record before you can assign a value to it.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to assign the variable.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); //Might throw exception if message doesn't exist or not visible due to scope.
sm.setStringParameter("s","NOW");
```

`RESTMessageV2 - setStringParameterNoEscape(String name, String value)`

Sets a REST message function variable with the specified name from the REST message record to the specified value.

This method is equivalent to `setStringParameter` but does not escape XML reserved characters.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the REST message variable. This parameter must be defined in the REST message record before you can assign a value to it.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to assign the variable.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

In the following example, replace `REST_message_record` with the name of the REST message record from your instance.

```javascript
var sm = new sn_ws.RESTMessageV2("<REST_message_record>","get"); //Might throw exception if message doesn't exist or not visible due to scope.
sm.setStringParameterNoEscape("s","NOW");
```
RESTResponseV2

The RESTResponseV2 API allows you to use the data returned by an outbound REST message in JavaScript code.

A RESTResponseV2 object is returned by the RESTMessageV2 functions execute() and executeAsync().

You can use this API in scoped applications, or within the global scope.

RESTResponseV2 - getAllHeaders()

Returns all headers contained in the response, including any duplicate headers.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List&lt;GlideHTTPHeader&gt;</td>
<td>The list of headers contained in the response. Each header is represented as a GlideHTTPHeader object which contains the header name and value.</td>
</tr>
</tbody>
</table>

```javascript
var r = new sn_ws.RESTMessageV2('<A REST message>', 'get');
var response = r.execute();
var headers = response.getAllHeaders();
for(var i in headers){
  gs.print(headers[i].name + ': ' + headers[i].value);
}
```

RESTResponseV2 - getBody()

Get the content of the REST response body.

Use this function when you want to get the request body as text content. Do not use this method when saving the response as a binary attachment. If a RESTMessageV2 object called the saveResponseBodyAsAttachment(...) function, using getBody() on the associated RESTResponseV2 object will cause an error. When saving the response as an attachment, if the outbound REST message fails, call getErrorMessage() on the response to retrieve the body content.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The REST response body.</td>
</tr>
</tbody>
</table>

```
var sm = new sn_ws.RESTMessageV2("Yahoo Finance","get"); // Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.execute();
var responseBody = response.getBody();
```

**RESTResponseV2 - getCookies()**

Returns all cookies included in the response.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The list of cookies. Iterate through the list to perform operations on each cookie.</td>
</tr>
</tbody>
</table>

```
var cookies = response.getCookies();
var i;
for(i=0;i<cookies.size();i++) {
    gs.print('cookie: ' + cookies.get(i));
}
```

Output:
- JSESSIONID=4135AA97A5D12DA22EF614AA2B0CAFDb.node20; Path=/; Secure; HttpOnly
- SABASESSIONID=370152970.36895.0000; path=/

**RESTResponseV2 - getErrorCode()**

Get the numeric error code if there was an error during the REST transaction. This error code is specific to the Now Platform, it is not an HTTP error code. Provide this error code if you require assistance from ServiceNow Customer Support.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The numeric error code, such as 1 for socket timeout.</td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.RESTMessageV2("Yahoo Finance","get"); //Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.execute();
var errorCode = response.getErrorCode();
```

**RESTResponseV2 - getErrorMessage()**

Get the error message if there was an error during the REST transaction.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The error message.</td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.RESTMessageV2("Yahoo Finance","get"); //Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.execute();
var errorMsg = response.getErrorMessage();
```

**RESTResponseV2 - getHeader(String name)**

Get the value for a specified header.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the header that you want the value for, such as <code>Set-Cookie</code>.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the specified header.</td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.RESTMessageV2("Yahoo Finance","get"); //Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.execute();
var headerVal = response.getHeader("Content-Type");
```

**RESTResponseV2 - getHeaders()**

Get all headers returned in the REST response and the associated values.

⚠️ **Note:** If a header is present more than once in the response, such as a `Set-Cookie` header, this function returns only the last of the duplicate headers. To return all headers including duplicates, use the `getAllHeaders()` function.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An Object that maps the name of each header to the associated value.</td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.RESTMessageV2("Yahoo Finance","get"); //Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.execute();
var headers = response.getHeaders();
```

### RESTResponseV2 - getQueryString()

Get the fully-resolved query sent to the REST endpoint.
This query contains the endpoint URL as well as any values assigned to variables in the REST message. Use this method only with responses to direct requests. This method is not supported for requests sent asynchronously, or requests sent using a MID server.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The fully-resolved query.</td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.RESTMessageV2("Yahoo Finance","get"); // Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.execute();
var queryString = response.getQueryString();
```

### RESTResponseV2 - getResponseAttachmentSysid()

Get the sys_id value of the attachment created from the response body content.

If the RESTMessageV2 object associated with this response called the saveResponseBodyAsAttachment(...) function, use getResponseAttachmentSysid() to get the sys_id of the created attachment record. Use this function when you want to perform additional operations with the new attachment record.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the new attachment record.</td>
</tr>
</tbody>
</table>

### RESTResponseV2 - getStatusCode()

Get the numeric HTTP status code returned by the REST provider.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The numeric status code returned by the REST provider, such as 200 for a successful response.</td>
</tr>
</tbody>
</table>

var sm = new sn_ws.RESTMessageV2("Yahoo Finance","get"); // Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.execute();
var statusCode = response.getStatusCode();

RESTResponseV2 - haveError()

Indicate if there was an error during the REST transaction.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Returns true if there was an error, false if there was no error.</td>
</tr>
</tbody>
</table>

var sm = new sn_ws.RESTMessageV2("Yahoo Finance","get"); // Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.execute();
var error = response.haveError();

RESTResponseV2 - waitForResponse(Number timeoutSecs)

Set the amount of time the instance waits for a response from the web service provider.
This method overrides the property glide.rest.outbound.ecc_response.timeout for this REST response.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeoutSecs</td>
<td>Number</td>
<td>The amount of time, in seconds, to wait for this response.</td>
</tr>
</tbody>
</table>
ServiceNow    Kingston    Now Platform Custom Business Applications

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
var sm = new sn_ws.RESTMessageV2("Yahoo Finance","get"); // Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.executeAsync();
response.waitForResponse(60);
```

RoundingInterval

Handles all the details of rounding intervals for CPU speed and RAM size. 
Use in any server-side Discovery script.

RoundingInterval - getRoundedValue(Number value)

Returns the value after applying the rounding interval.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Number</td>
<td>The value to round off</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The value after rounding</td>
</tr>
</tbody>
</table>

RoundingInterval - RoundingInterval(String type)

Creates an instance of the RoundingInterval class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Can be either CPU or RAM.</td>
</tr>
</tbody>
</table>

ScriptLoader

Provides the ability to load scripts asynchronously.

You can use the ScriptLoader API in client-side scripts for a platform/desktop UI using ListV2 and ListV3 APIs. It is not available for Service Portal, Mobile, or Agent Workspace.

You access the ScriptLoader methods by using the global object ScriptLoader.
**ScriptLoader - getScripts(String filePath, Function callback)**

Gets scripts asynchronously.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filePath</td>
<td>String</td>
<td>A path, including the file name, that contains one or more scripts.</td>
</tr>
<tr>
<td>callback</td>
<td>Function</td>
<td>The function to be called after the scripts have been loaded. The callback function has no arguments.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ScriptLoader - getScripts(Array scripts, Function callback)**

Loads scripts asynchronously.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scripts</td>
<td>Array</td>
<td>An array of scripts.</td>
</tr>
<tr>
<td>callback</td>
<td>Function</td>
<td>The function called when the scripts have been loaded. The callback function has no arguments.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**SerialNumberManager**

Manages the serial numbers for discovery devices.

Use with a discovery script where you need to manage serial numbers.

**SerialNumberManager - add(Number sType, Number value)**

Adds the specified serial number.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sType</td>
<td>Number</td>
<td>The serial number type</td>
</tr>
<tr>
<td>value</td>
<td>Number</td>
<td>The serial number</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

SerialNumberManager - get()

Returns the serial number.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The serial number</td>
</tr>
</tbody>
</table>

SerialNumberManager - getSerialsForCIData()

Returns serial number information for CI Data, including the type, serial number, and validity.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array with the serial number information in the following order: serial number type, serial number, and validity flag.</td>
</tr>
</tbody>
</table>

SerialNumberManager - isValid(Number value)

Checks if the number is a valid serial number.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Number</td>
<td>The number to check</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the number is valid; otherwise, false.</td>
</tr>
</tbody>
</table>

**ServiceRegistryQuery**

Maps Shazzam service registry query results against a port probe. Child classes provide specifics on mapping.

Use this API during the discovery scanning phase.

**ServiceRegistryQuery - findForPortProbe(PortProbe portProbe)**

Parses an array of IDs for the given port probe.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>portProbe</td>
<td>PortProbe</td>
<td>The port probe</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ServiceRegistryQuery - get(Object source)**

Retrieves a registry service by id, regardless of child class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Object</td>
<td>The registry service GlideRecord or sys_id.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The registry service</td>
</tr>
</tbody>
</table>

**ServiceRegistryQuery - getShazzamQuerierClassname()**

Returns the Shazzam! querier class name.
### ServiceRegistryQuery - toPortProbeMap()

Retrieves the port probe XML and creates a map.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The class name</td>
</tr>
</tbody>
</table>

### ServiceRegistryQuery - toPortProbeXml()

Creates XML output for Shazzam! port probes.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The port probe map</td>
</tr>
</tbody>
</table>

### ServiceRegistryQuery - ServiceRegistryQuery(Object source)

Creates an instance of the ServiceRegistryQuery class.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Object</td>
<td>A GlideRecord or sys_id of the registry service.</td>
</tr>
</tbody>
</table>
ShazzamLaunch
Handles the details of launching a Shazzam probe for a given discovery.
Use this API to launch a Shazzam probe during the discovery scanning phase.

ShazzamLaunch - addPortProbe(String midPPs, String portProbes, IPIncludeExcludeCollection ipiec)

Adds a single port probe to our MID services collection.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>midPPs</td>
<td>String</td>
<td>A hashmap with two properties.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• midServer: a MIDServer instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• portProbes: a hashmap (by port probe name) of DiscoveryPortProbe instances.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>portProbes</td>
<td>String</td>
<td>A hashmap (by MIDServer name) that contains these properties.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• range: An IPMetaCollection containing the ranges to be probed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• midServer: A MIDServer instance for the MID server to be probed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• portProbes: a hashmap (by port probe name) of DiscoveryPortProbe instances to be probed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipiec</td>
<td>IPIncludeExcludeCollection</td>
<td>The range to be probed.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

ShazzamLaunch - addRange(IPCollection ipc, String portProbes, Boolean samePhase)

Adds the specified range and this discovery's port probes to scan (as defined by behavior) to the collection of MID servers and port probes that need to be probed.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipc</td>
<td>IPCollection</td>
<td>The range to be discovered</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>portProbes</td>
<td>String</td>
<td>A hashmap (by MID server name) of hashmaps containing these properties.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• range: an IPMetaCollection containing the ranges to be probed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• midServer: a MID server instance for the MID server to be probed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• portProbes: a hashmap (by port probe name) of DiscoveryPortProbe instances to be probed.</td>
</tr>
<tr>
<td>samePhase</td>
<td>Boolean</td>
<td>True if this should use the same phase as is currently running.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**ShazzamLaunch - fireShazzamProbe(String portProbe, String ip, String sensor)**

Fires a Shazzam probe for the MID server and services in the given hash map.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>portProbe</td>
<td>String</td>
<td>A hashmap containing these properties.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• range: an IPMetaCollection containing the ranges to be probed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• midServer: a MIDServer instance for the MID server to be probed.</td>
</tr>
<tr>
<td>ip</td>
<td>String</td>
<td>A comma-separated list of IPs to fire probes for.</td>
</tr>
<tr>
<td>sensor</td>
<td>String</td>
<td>The sys_id of the sensor that launched this probe (network discovery only).</td>
</tr>
</tbody>
</table>
### ShazzamLaunch - launch(String ip, Boolean samePhase, String sensor)
Launches Shazzam probes as necessary, depending on the current phase of each behavior used.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>String</td>
<td>A comma-separated list of IPs to fire probes for.</td>
</tr>
<tr>
<td>samePhase</td>
<td>Boolean</td>
<td>True if this launch should use the same phase as is currently running.</td>
</tr>
<tr>
<td>sensor</td>
<td>String</td>
<td>The sys_id of the sensor that launched this (only in network discovery).</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if any probes are launched.</td>
</tr>
</tbody>
</table>

### ShazzamLaunch - makeProbeSpec(String portProbe)
Makes an XML port probe spec for the given port probe hashmap.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>portProbe</td>
<td>String</td>
<td>A hashmap containing these properties.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• range: an IPMetaCollection containing the ranges to be probed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• midServer: a MIDS der server instance for the MID server to be probed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• portProbes: a hashmap (by port probe name) of DiscoveryPortProbe instances to be probed.</td>
</tr>
</tbody>
</table>
## ShazzamLaunch

**ShazzamLaunch(DiscoveryStatus status, DiscoverySchedule schedule)**

Creates an instance of the ShazzamLaunch class.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>DiscoveryStatus</td>
<td>The DiscoveryStatus instance for this discovery.</td>
</tr>
<tr>
<td>schedule</td>
<td>DiscoverySchedule</td>
<td>The DiscoverySchedule instance for this discovery.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

## ShazzamResult

Encapsulates a Shazzam result.

Use this API during the discovery scanning phase.

### ShazzamResult - active

Property that shows if the IP address is active

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

True if this IP address is active, which means at least one port is open.

### ShazzamResult - alive

Property is true if this IP address is alive, which means no ports are open, but at least one responded.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alive</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

True if the IP address is alive, meaning no ports are open, but at least one port responded.
**ShazzamResult - domainName**

The Windows domain name for the IP.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domainName</td>
<td>String</td>
<td>The Windows domain name for the IP if one was resolved by the scanner.</td>
</tr>
</tbody>
</table>

**ShazzamResult - hostName**

Host name for the IP.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostName</td>
<td>String</td>
<td>Host name of the IP if one was resolved by the scanner.</td>
</tr>
</tbody>
</table>

**ShazzamResult - ip**

IP address of the results.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>String</td>
<td>IP address of the results</td>
</tr>
</tbody>
</table>

**ShazzamResult - scanners**

A JavaScript array of ShazzamScanner instances.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scanners</td>
<td>Array</td>
<td>Array of ShazzamScanner instances</td>
</tr>
</tbody>
</table>

**ShazzamScanner**

Encapsulates a Shazzam scanner.

Use this API during the discovery scanning phase.

**ShazzamScanner - contents**

A hash map of any nodes contained within the scanner, by name.
<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contents</td>
<td>Hash map</td>
<td>A hash map of any nodes contained within the scanner, by name.</td>
<td></td>
</tr>
</tbody>
</table>

**ShazzamScanner - domainName**
The Windows domain name, if one was resolved by the scanner.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domainName</td>
<td>String</td>
<td>The Windows domain name, if one was resolved by the scanner.</td>
<td></td>
</tr>
</tbody>
</table>

**ShazzamScanner - hostName**
The IP hostname, if one was resolved by the scanner.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostName</td>
<td>String</td>
<td>The IP hostname, if one was resolved by the scanner.</td>
<td></td>
</tr>
</tbody>
</table>

**ShazzamScanner - name**
The scanner name.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The scanner name</td>
<td></td>
</tr>
</tbody>
</table>

**ShazzamScanner - port**
The port being scanned.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>String</td>
<td>The port being scanned.</td>
<td></td>
</tr>
</tbody>
</table>

**ShazzamScanner - portProbe**
The port probe name.
**Field**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>portProbe</td>
<td>String</td>
<td>The port probe name.</td>
</tr>
</tbody>
</table>

**ShazzamScanner - protocol**

The scanner protocol.

**Field**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>String</td>
<td>The scanner protocol.</td>
</tr>
</tbody>
</table>

**ShazzamScanner - result**

The scanner result.

**Field**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>String</td>
<td>The scanner result.</td>
</tr>
</tbody>
</table>

**ShazzamScanner - service**

The scanner service name.

**Field**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>String</td>
<td>The scanner service name.</td>
</tr>
</tbody>
</table>

**SLARepair**

The SLARepair API first deletes the existing SLAs and then recreates them from each task’s history.

**SLARepair - repairByFilter(String filter, String sourceTable)**

Repair the task SLAs associated with the passed-in filter and source table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>string</td>
<td>Specify an encoded query that is used to retrieve a set of records from the source table.</td>
</tr>
<tr>
<td>sourceTable</td>
<td>string</td>
<td>Specify the name of a table that is (or extends) contract_sla, task_sla, or task.</td>
</tr>
</tbody>
</table>
ServiceNow    Kingston    Now Platform Custom Business Applications

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Repair SLAs for problems created last month with a priority of 2.

```javascript
var gr = new GlideRecord("problem");
gr.addQuery("sys_created_on", "ON", "Last Month@javascript:gs.beginningOfLastMonth()@javascript:gs.endOfLastMonth()");
gr.addQuery("priority", "2");
gr.query();

var repair = new SLARepair();
while (gr.next())
    repair.repairByFilter(gr.getEncodedQuery(),
                          gr.getRecordClassName());
```

SLARepair - repairByGlideRecord(GlideRecord gr)
Repair the task SLAs associated with the passed in GlideRecord.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>GlideRecord</td>
<td>Specify a GlideRecord for a table that is (or extends) contract_sla, task_sla, or task.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Repair SLAs for problems created last month with a priority of 2.

```javascript
var gr = new GlideRecord("problem");
gr.addQuery("sys_created_on", "ON", "Last Month@javascript:gs.beginningOfLastMonth()@javascript:gs.endOfLastMonth()");
gr.addQuery("priority", "2");
gr.query();

var repair = new SLARepair();
while (gr.next())
    repair.repairByGlideRecord(gr);
```

SLARepair - repairBySysId(String sysId, String sourceTable)
Repair the task SLAs associated with the passed in sys_id and source table.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysId</td>
<td>string</td>
<td>Specify the ID of a table that is (or extends) contract_sla, task_sla, or task.</td>
</tr>
<tr>
<td>sourceTable</td>
<td>string</td>
<td>Specify the name of a table that is (or extends) contract_sla, task_sla, or task.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Find problems created last month with a priority of 2

```javascript
var gr = new GlideRecord("problem");
gr.addQuery("sys_created_on", "ON", "Last Month@javascript:gs.beginningOfLastMonth()@javascript:gs.endOfLastMonth()" );
gr.addQuery("priority", "2");
gr.query();

var repair = new SLARepair();
while (gr.next())
    repair.repairBySysId(gr.sys_id + ",
                         gr.getRecordClassName());
```

### SLARepair - setAuditEnabled(Boolean onOrOff)

Enables or disables auditing when running a repair.

By default, auditing is set to the value in the property `com.snc.sla.repair.audit`. You can override this with passing in true to enable or false to disable auditing.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOrOff</td>
<td>Boolean</td>
<td>Determines whether to enable or disable auditing.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>A self-reference to allow for method chaining.</td>
</tr>
</tbody>
</table>

```javascript
var builder = new SLARepair();
```
SLARepair - `setRunWorkflow(Boolean onOrOff)`

Enables or disables running a workflow for each of the Task SLA records being repaired.

By default, when a Task SLA is repaired the workflow will be run during the repair process. To override this, you can pass in false to disable running of the workflow or true to enable it.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOrOff</td>
<td>Boolean</td>
<td>Determines whether to enable or disable workflow.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>A self-reference to allow for method chaining.</td>
</tr>
</tbody>
</table>

```javascript
var repair = new SLARepair();
repair.setRunWorkflow(false);
```

SLARepair - `setValidateOnly(Boolean onOrOff)`

Validates the repair request.

If false is passed in, the task SLAs will be repaired. If true is passed in, calls to repair will not alter any task SLAs but only validate the supplied parameters and generate a count of records to be repaired.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onOrOff</td>
<td>Boolean</td>
<td>Determines whether to enable or disable validation.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>A self-reference to allow for method chaining.</td>
</tr>
</tbody>
</table>

```javascript
var repair = new SLARepair();
repair.setValidateOnly(true);
```
**SLPServiceRegistryQuery**
Maps Shazzam! SLP query results to port probes.
Use this API during the discovery scanning phase.

**SLPServiceRegistryQuery - getShazzamQuerierClassname()**
Returns the Shazzam querier class name.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>String</td>
</tr>
</tbody>
</table>

**SLPServiceRegistryQuery - toPortProbeMap()**
Creates a scalar map for port probes to use in XML output to Shazzam!.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Object</td>
</tr>
</tbody>
</table>

**SNMPGatherDocParts**
Handles gathering the parts of an SNMP response document for JavaScript SNMP sensors.
Use with a discovery script when you need to gather the parts of an SNMP response document.

**SNMPGatherDocParts - gather(String doc, Object probe)**
Returns the given XML document with any data gathered by prior SNMP probes.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>doc</td>
</tr>
<tr>
<td>probe</td>
</tr>
</tbody>
</table>
## SNMPNetworkInterfaces

Creates network interfaces for JavaScript SNMP sensors.

Use with a discovery script when you need to create network interfaces for JavaScript SNMP sensors.

**SNMPNetworkInterfaces - process(String cmdb_ci, String doc, Boolean isComputer, String agentName, String ecc_sys_id)**

Analyzes the SNMP data and reconciles the discovered NICs to the CMDB's NICs.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmdb_ci</td>
<td>String</td>
<td>The sys_id of the CI</td>
</tr>
<tr>
<td>doc</td>
<td>String</td>
<td>The SNMP data in XML form</td>
</tr>
<tr>
<td>isComputer</td>
<td>Boolean</td>
<td>If true, this is a computer; otherwise, false.</td>
</tr>
<tr>
<td>agentName</td>
<td>String</td>
<td>Name of the MID server</td>
</tr>
<tr>
<td>ecc_sys_id</td>
<td>String</td>
<td>The sys_id of the ECC queue record with the input.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

## SNMPResponse

Wraps an SNMP payload response instance with methods to safely and easily retrieve SNMP singleton fields or tables.

Use to retrieve SNMP singleton fields or tables.

**SNMPResponse - buildIndex(Object cur, String name, String extName)**

Builds an index to all SNMP entities in the current object.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cur</td>
<td>Object</td>
<td>The current object</td>
</tr>
</tbody>
</table>
### ServiceNow Custom Business Applications

#### Name | Type | Description
--- | --- | ---
name | String | The index name
extName | String | The index extension

### SNMPResponse - buildReferences()

Builds references to actual instances for all identified indexes (in the discovery_snmp_ref table) that can be resolved.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SNMPResponse - getAmbiguousOIDs(String refBase, String refLeaf)

Returns an array of OIDs that match the given ref_base and which have a defined, non-null ref_leaf.

The last character of ref_base may be a question mark; in that case, if the parent node is a table then all table entries are returned.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| refBase| String  | The reference base
| refLeaf| String  | The reference leaf

### SNMPResponse - getOIDInt(String name)

Returns the integer contained in the OID object with the given MIB name.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The MIB name</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The integer contained in the OID object, or zero if the object could not be found or is empty.</td>
</tr>
</tbody>
</table>

#### SNMPResponse - getOIDObj(String name)

Retrieves the OID object with the given MIB name from the response.

This method does not work with numeric OID strings, only MIB names.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The MIB name</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OID object</td>
<td>The OID object with the given MIB name, or null if the object could not be found or if any table entries are encountered while walking down the named elements.</td>
</tr>
</tbody>
</table>

#### SNMPResponse - getOIDTable(String parentName, String entryName)

Returns a hashmap of table entries under the given parent MIB name, with the given entry MIB names.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentName</td>
<td>String</td>
<td>The parent MIB name</td>
</tr>
<tr>
<td>entryName</td>
<td>String</td>
<td>The entry MIB name</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hashmap</td>
<td>A hashmap of table entries under the given parent MIB name, with the given entry MIB names, or null if the parent does not exist.</td>
</tr>
</tbody>
</table>
**SNMPResponse - getOIDText(String name)**

Returns the text contained in the OID object with the given MIB name.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The MIB name</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The text contained in the OID object. Returns null if the object could not be found, or if it has no text.</td>
</tr>
</tbody>
</table>

**SNMPResponse - normalize(Object cur)**

Resolves SNMP tables into hashmaps and OID values into primitives.

All SNMP numeric types convert to a JavaScript number. An SNMP null type converts to a JavaScript null. An SNMP IP Address value converts to a Java IPAddressV4 instance. All other values convert to a JavaScript string.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cur</td>
<td>Object</td>
<td>The SNMP table or OID value.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**SOAPMessageV2**

The SOAPMessageV2 API allows you to send an outbound SOAP message using JavaScript.

Use the SOAPResponseV2 API to manage the response returned by the SOAP provider.

You can use this API in scoped applications, or within the global scope.

**SOAPMessageV2 - execute()**

Send the SOAP message to the endpoint.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SOAPResponseV2

The response returned by the SOAP provider.

### SOAPMessageV2 - executeAsync()

Send the SOAP message to the ECC queue.

SOAP messages in the ECC queue are processed by the SOAPClient business rule.

By default, this business rule does not run asynchronously. To configure this business rule to run asynchronously, set the `When` value to `Async` and add `current.update()` to the end of the `Script`. The instance does not wait for a response from the web service provider when sending a message through the ECC queue.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAPResponseV2</td>
<td>The response returned by the SOAP provider.</td>
</tr>
</tbody>
</table>

**Note:** Attempting to use the SOAP response object before the response has been processed may result in a timeout error.

```javascript
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.execute(); // Might throw exception if HTTP connection timed out or some issue with sending request itself because of encryption/decryption of password.
```

### SOAPMessageV2 - getEndpoint()

Get the endpoint for the SOAP message.

```javascript
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
var response = sm.executeAsync();
```
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The URL of the SOAP web service provider.</td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
var endpoint = sm.getEndpoint();
```

**SOAPMessageV2 - getRequestBody()**

Get the content of the SOAP message body.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The SOAP message body.</td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.execute();
var requestBody = sm.getRequestBody();
```

**SOAPMessageV2 - getRequestHeader(String headerName)**

Get the value for an HTTP header specified by the SOAP client.

By default, this method cannot return the value for a header set automatically by the system. To grant this method access to all headers, set the property glide.http.log_debug to true.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headerName</td>
<td>String</td>
<td>The request header you want to get the value for.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the specified header.</td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
var header = sm.getRequestHeader("Accept");
```

**SOAPMessageV2 - getRequestHeaders()**

Get HTTP headers that were set by the SOAP client and the associated values.

This method does not return headers set automatically by the system. To configure this method to return all headers, set the property glide.http.log_debug to true.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An Object that maps the name of each header to the associated value.</td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
var requestHeaders = sm.getRequestHeaders();
```

**SOAPMessageV2 - setBasicAuth(String userName, String userPass)**

Set basic authentication headers for the SOAP message.

Setting basic authentication headers using this method overrides basic authentication values defined in the SOAP message record.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userName</td>
<td>String</td>
<td>The username to use when authenticating the SOAP message.</td>
</tr>
<tr>
<td>userPass</td>
<td>String</td>
<td>The password for the specified user.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setBasicAuth("username","password");
```

**SOAPMessageV2 - setEccCorrelator(String correlator)**

Associate outbound requests and the resulting response record in the ECC queue.

This method only applies to SOAP messages sent through a MID Server. The correlator provided populates the **Agent correlator** field on the ECC queue record for the response. Provide a unique correlator for each outbound request to associate the correct results in the ECC queue with the request when designing asynchronous automation through a MID Server.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>String</td>
<td>A unique identifier.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setEccCorrelator("unique_id");
```

**SOAPMessageV2 - setEccParameter(String name, String value)**

Override a value from the database by writing to the SOAP message payload.
This method only applies to SOAP messages sent through a MID Server. Use this method when a value from the SOAP message in the database is invalid, such as when the endpoint URL is longer than the maximum SOAP endpoint field length.

These are valid values for the name parameter.
- **source**: the endpoint URL.
- **name**: the SOAP message function to run.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the ECC parameter.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to assign to the specified ECC parameter.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setEccParameter("source","http://very.long.endpoint");
```

### SOAPMessageV2 - setEndpoint(String endpoint)

Set the endpoint for the SOAP message.

By default, the SOAP message uses the endpoint specified in the SOAP message record. Use this method to override the default. You must call this method when using the **SOAPMessageV2** constructor with no parameters.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
<td>String</td>
<td>The URL of the SOAP web service provider you want to interface with.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.SOAPMessageV2();
sm.setEndpoint("http://web.service.endpoint");
```

**SOAPMessageV2 - setHttpTimeout(Number timeoutMs)**

Set the amount of time the SOAP message waits for a response from the web service provider before the request times out.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeoutMs</td>
<td>Number</td>
<td>The amount of time to wait for a response from the web service provider, in milliseconds.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setHttpTimeout(6000);
```

**SOAPMessageV2 - setLogLevel(String level)**

Sets the log level for this message and the corresponding response.

Setting a log level using the SOAPMessageV2 API overrides the log level configured on the SOAP message record. This log level may not apply if the endpoint domain is blacklisted, or if the property glide.outbound_http_log.override is true. To view outbound web service logs, navigate to System Logs > Outbound HTTP Requests.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>level</td>
<td>String</td>
<td>The log level. Valid values are basic, elevated, and all.</td>
</tr>
</tbody>
</table>
SOAPMessageV2 - setMIDServer(String midServerName)

Configure the SOAP message to be sent through a MID Server. By default, the SOAP message uses the MID Server specified in the SOAP message function record. Use this method to override the default.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>midServerName</td>
<td>String</td>
<td>The name of the MID Server you want to send the SOAP message through. Your instance must have an active MID Server with the specified name.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setMutualAuth("auth_profile_name");
SOAPMessageV2 - setRequestBody(String requestBody)

Set the body content to send to the web service provider.

When you set the body content using this method, variables in the body are not substituted for parameters from the SOAP message function record. You must explicitly define all values within the SOAP message body.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestBody</td>
<td>String</td>
<td>The body of the SOAP message.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
var body = "<SOAP message body>";
sm.setRequestBody(body);
```

SOAPMessageV2 - setRequestHeader(String headerName, String headerValue)

Set an HTTP header in the SOAP message to the specified value.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headerName</td>
<td>String</td>
<td>The name of the header.</td>
</tr>
<tr>
<td>headerValue</td>
<td>String</td>
<td>The value to assign to the specified header.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setRequestHeader("Accept","Application/json");
```
**SOAPMessageV2 - setSOAPAction(String soapAction)**

Define the SOAP action this SOAP message performs.

The WSDL for your web service provider lists SOAP actions you can perform. You must call this method when using the `SOAPMessageV2()` constructor with no parameters.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>soapAction</td>
<td>String</td>
<td>The SOAP action this SOAP message performs.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var sm = new sn_ws.SOAPMessageV2();
sm.setSOAPAction("GetQuote");
//construct SOAP message by specifying endpoint and auth
sm.execute();
```

**SOAPMessageV2 - setStringParameter(String name, String value)**

Set a variable with the specified name from the SOAP message record to the specified value. XML reserved characters in the value are converted to the equivalent escaped characters.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the SOAP message variable.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to assign to the specified variable.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setStringParameter("symbol","NOW");
```
SOAPMessageV2 - setStringParameterNoEscape(String name, String value)

Set a variable with the specified name from the SOAP message record to the specified value. This method is equivalent to setStringParameter but does not escape XML reserved characters.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the SOAP message variable.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to assign to the specified variable.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setStringParameterNoEscape("symbol","NOW");
```

SOAPMessageV2 - setWSSecurity(String keystoreId, String keystoreAlias, String keystorePassword, String certificateId)

Sets web service security values for the SOAP message.

Setting security values using this method overwrites web service security values defined for the SOAP message record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>keystoreId</td>
<td>String</td>
<td>The sys_id of the Java or PKCS12 key store to use.</td>
</tr>
<tr>
<td>keystoreAlias</td>
<td>String</td>
<td>The alias that identifies the public and private keys.</td>
</tr>
<tr>
<td>keystorePassword</td>
<td>String</td>
<td>The password assigned to the key store record.</td>
</tr>
<tr>
<td>certificateId</td>
<td>String</td>
<td>The sys_id of the trusted server certificate.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
sm.setWSSecurity("70d65e074f3812001f6eac118110c71a","Quote keys","UXr82cqX75Z7MaSa+EyjGA==","ba969a074f3812001f6eac118110c76d");
```

SOAPMessageV2 - SOAPMessageV2()

Instantiates an empty SOAPMessageV2 object.

When using an object instantiated this way, you must manually specify a SOAP action and endpoint.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
var sm = new sn_ws.SOAPMessageV2();
```

SOAPMessageV2 - SOAPMessageV2(String soapMessage, String soapFunction)

Instantiate a SOAPMessageV2 object from a SOAP message record and a function associated with that record.

Values such as the endpoint, authentication, or MID Server settings from the SOAP message record apply to this object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>soapMessage</td>
<td>String</td>
<td>The SOAP message record you want to use as the base for this object.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>soapFunction</td>
<td>String</td>
<td>The SOAP function you want to execute. Available SOAP functions depend on the WSDL supplied by the web service provider.</td>
</tr>
</tbody>
</table>

```javascript
var sm = new sn_ws.SOAPMessageV2("StockQuote","GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope.
```

**SOAPResponseV2**

The SOAPResponseV2 API allows you to use the data returned by an outbound SOAP message in JavaScript code.

A SOAPResponseV2 object is returned by the SOAPMessageV2 functions execute() and executeAsync().

You can use this API in scoped applications, or within the global scope.

**SOAPResponseV2 - getAllHeaders()**

Return all headers contained in the response, including any duplicate headers.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List&lt;GlideHTTPHeader&gt;</td>
<td>The list of headers contained in the response. Each header is represented as a GlideHTTPHeader object which contains the header name and value.</td>
</tr>
</tbody>
</table>

```javascript
var r = new sn_ws.SOAPMessageV2('<A SOAP message>', 'get');
var response = r.execute();
var headers = response.getAllHeaders();
for(var i in headers){
  gs.print(headers[i].name + ': ' + headers[i].value);
}
```

**SOAPResponseV2 - getBody()**

Get the content of the SOAP response body.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The SOAP response body.</td>
</tr>
</tbody>
</table>

```javascript
var body = response.getBody();
```

**SOAPResponseV2 - get Cookies()**

Returns all cookies included in the response.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The list of cookies. Iterate through the list to perform operations on each cookie.</td>
</tr>
</tbody>
</table>

```javascript
var cookies = response.getCookies();
var i;
for(i=0;i<cookies.size();i++) {
    gs.print('cookie: ' + cookies.get(i));
}
```

Output:
- JSESSIONID=4135AA97A5D12DA22EF614AA2B0CAF8.node20; Path=/; Secure; HttpOnly
- SABASESSIONID=370152970.36895.0000; path=/

**SOAPResponseV2 - get ErrorCode()**

Get the numeric error code if there was an error during the SOAP transaction.

This error code is specific to the Now Platform, it is not an HTTP error code. Provide this error code if you require assistance from ServiceNow Technical Support.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The numeric error code, such as 1 for a socket timeout.</td>
</tr>
</tbody>
</table>

```javascript
var errorCode = response.getErrorCode();
```

**SOAPResponseV2 - getErrorMessage()**

Get the error message if there was an error during the SOAP transaction.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The error message.</td>
</tr>
</tbody>
</table>

```javascript
var errorMsg = response.getErrorMessage();
```

**SOAPResponseV2 - getHeader(String name)**

Get the value for a specified HTTP header.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name of the header that you want the value for, such as Set-Cookie.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the specified header.</td>
</tr>
</tbody>
</table>

```
var headerVal = response.getHeader("Accept");
```

**SOAPResponseV2 - getHeaders()**

Get all HTTP headers returned in the SOAP response and the associated values.

**Note:** If a header is present more than once in the response, such as a Set-Cookie header, this function returns only the last of the duplicate headers. To return all headers including duplicates, use the `getAllHeaders()` function.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An Object that maps the name of each header to the associated value.</td>
</tr>
</tbody>
</table>

```
var headers = response.getHeaders();
```

**SOAPResponseV2 - getStatusCode()**

Get the numeric HTTP status code returned by the SOAP provider.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The numeric status code returned by the SOAP provider, such as 200 for a successful response.</td>
</tr>
</tbody>
</table>

```javascript
var statusCode = response.getStatusCode();
```

**SOAPResponseV2 - haveError()**

Indicate if there was an error during the SOAP transaction.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Returns true if there was an error, false if there was no error.</td>
</tr>
</tbody>
</table>

```javascript
var error = response.haveError();
```

**SOAPResponseV2 - waitForResponse(Number timeoutSecs)**

Set the amount of time the instance waits for a response from the web service provider. This method overrides the property glide.soap.outbound.ecc_response.timeout for this SOAP response.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeoutSecs</td>
<td>Number</td>
<td>The amount of time, in seconds, to wait for this response.</td>
</tr>
</tbody>
</table>
**spAriaUtil**

You can show messages on a screen reader. The spAriaUtil service is an angular service included as part of the Service Portal angular application. The spAriaUtil service is available in the client script block of Service Portal widgets.

**spAriaUtil - sendLiveMessage(String message)**

Announce a message to a screen reader.

The `sendLiveMessage()` method injects text into an aria-live region on the page. The default setting for an aria-live region is `assertive`, which means that messages are announced immediately. This can annoy and confuse users if used too frequently.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to be shown.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
function(spAriaUtil) {
    /* widget controller */

    spAriaUtil.sendLiveMessage('Hello world!');
}
```

**spModal**

The spModal class provides an alternative way to show alerts, prompts, and confirmation dialogs. The SPModal class is available in Service Portal client scripts.

You can use `spModal.open()` to display a widget in a modal dialog. The spModal class is a lightweight wrapper for Angular UI bootstrap’s `SubModal`.

**spModal - alert(String message).then(fn)**

Displays an alert.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to display.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>The promise contains a single argument that contains true or false.</td>
</tr>
</tbody>
</table>

/// HTML template
<button ng-click="c.onAlert()" class="btn btn-default">Alert</button>

/// Client script
function(spModal) {
  var c = this;
  c.onAlert = function () {
    spModal.alert('How do you feel today?').then(function (answer) {
      c.simple = answer;
    });
  };
}

spModal - confirm(String message).then(fn)

Displays a confirmation message.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to display.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>The promise contains a single argument that contains true or false.</td>
</tr>
</tbody>
</table>

/// HTML template
<button ng-click="c.onConfirm()" class="btn btn-default">Confirm</button>
<span>{{c.confirmed}}</span>

/// Client script
function(spConfirmed) {

```javascript
var c = this;
c.onConfirm = function() {
  c.confirmed = "asking";
  spModal.confirm("Can you confirm or deny this?").then(function(confirmed) {
    c.confirmed = confirmed; // true or false
  });
}
```

Confirm with HTML message

```html
//HTML template
<button ng-click="c.onConfirmEx()" class="btn btn-default">
  Confirm - HTML message
</button>
<span>{{c.confirmed}}</span>

// Client script
function(spModal) {
  var c = this;
  // more control, passing options
  c.onConfirmEx = function() {
    c.confirmed = "asking";
    var warn = '<i class="fa fa-warning" aria-hidden="true"></i>
    spModal.open({
      title: 'Delete this Thing?',
      message: warn + ' Are you <b>sure</b> you want to do that?'
    }).then(function(confirmed) {
      c.confirmed = confirmed;
    })
  }
}
```

**spModal - open(Object options).then(fn)**

Opens a modal window using the specified options.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>Object</td>
<td>An object that may have these properties.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- title - a string that can be HTML that goes in the header. The default is empty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- message - a string that can be HTML that goes in the header. The default is empty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- buttons - an array that contains the buttons to show on the dialog. The default is Cancel and OK.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- input - a Boolean. When true shows an input field on the dialog. The default is false.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- value - a string containing the value of the input field. The default is empty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- widget - a string that identifies the widget ID or sys_id to embed in the dialog. The default is empty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- widgetInput - an object to send the embedded widget as input. The default is null.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- shared - a client-side object to share data with the embedded widget client script.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- size - a string indicating the size of the window. Can be 'sm' or 'lg'. The default is empty.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example of a prompt with a label

```html
// HTML template
<button ng-click="c.onOpen()" class="btn btn-default">
  Prompt with label
</button>
>You answered <span>{{c.name}}</span></span>
```
//Client code
function(spModal) {
  var c = this;
  c.onOpen = function() {
    //ask the user for a string
    spModal.open({
      title: 'Give me a name',
      message: 'What would you like to name it?',
      input: true,
      value: c.name
    }).then(function(name) {
      c.name = name;
    })
  }
}

Example of agree with custom buttons.

//HTML template
<button ng-click="c.onAgree()" class="btn btn-default">
  Agree
</button>
<div ng-show="c.agree">
  You answered {{c.agree}}
</div>

//Client script
function(spModal) {
  var c = this;
  c.onAgree = function() {
    // ask the user for a string
    // note embedded html in message
    var h = '<h4>Apple likes people to agree to lots of stuff</h4>'
    // Line feeds added to the following lines for presentation formatting.
    var m = 'Your use of Apple software or hardware products is based on the software license and other terms and conditions in effect for the product at the time of purchase. Your agreement to these terms is required to install or use the product.'
    spModal.open({
      title: 'Do you agree?',
      message: h + m,
      buttons: [
        {label:'# ${No}', cancel: true},
        {label:'# ${Yes}', primary: true}
      ]
    }).then(function() {
      c.agree = 'yes';
    }, function() {
      c.agree = 'no';
    })
  }
}
Example of embedded widget

```html
// HTML template
<button ng-click="c.onWidget('widget-cool-clock')" class="btn btn-default">
  Cool Clock
</button>

// Client script
function(spModal) {
  var c = this;
  c.onWidget = function(widgetId, widgetInput) {
    spModal.open({
      title: 'Displaying widget ' + widgetId,
      widget: widgetId,
      widgetInput: widgetInput || {}
    }).then(function(){
      console.log('widget dismissed');
    })
  }
}
```

**spModal - prompt(String message, String default).then(fn)**

Displays a prompt for user input.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to display.</td>
</tr>
<tr>
<td>default (optional)</td>
<td>String</td>
<td>A default value to use if the user does not provide a response.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The promise contains the user's response, or the default value if the user does not enter a response.</td>
</tr>
</tbody>
</table>

```html
// HTML template
<button ng-click="c.onPrompt()" class="btn btn-default">
  Prompt
</button>
<div ng-show="c.name">
  You answered <span>{{c.name}}</span>
</div>

// Client script
function(spModal) {
  var c = this;
  c.onPrompt = function() {
    spModal.prompt('Display question', 'default').then(function(response) {
      console.log('User answered: ' + response);
    });
  }
}
```
spModal.prompt("Your name please", c.name).then(function(name) {
    c.name = name;
})

spUtil
Utility methods to perform common functions in a Service Portal widget client script.

spUtil - addErrorMessage(String message)
Displays a notification error message.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>Error message to display.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

spUtil.addErrorMessage("There has been an error processing your request")

spUtil - addInfoMessage(String message)
Displays a notification info message.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>Message to display.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

spUtil.addInfoMessage("Your order has been placed")
spUtil - addTrivialMessage(String message)

Displays a trivial notification message. Trivial messages disappear after a short period of time.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>Message to display.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

spUtil.addTrivialMessage("Thanks for your order")

spUtil - get(String widgetId Object data)

Returns a widget model by ID or sys_id. Use this method to embed a widget model in a widget client script. The callback function returns the full widget model.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>widgetId</td>
<td>String</td>
<td>Widget ID or sys_id of the widget to embed.</td>
</tr>
<tr>
<td>data</td>
<td>Object</td>
<td>(Optional) Name/value pairs of parameters to pass to the widget model.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Model of the embedded widget.</td>
</tr>
</tbody>
</table>

Without data passed

```javascript
spUtil.get("widget-cool-clock").then(function(response) {
  c.coolClock = response;
});
```
With data passed

```javascript
spUtil.get('pps-list-modal', {title: c.data.editAllocations,
  table: 'resource_allocation',
  queryString: 'GROUPBYuser^resource_plan=' + c.data.sysId,
  view: 'resource_portal_allocations' }).then(function(response) {
  var formModal = response;
  c.allocationListModal = response;
});
```

**spUtil - format(String template, Object data)**

Formats a string as an alternative to string concatenation. Use this method to build a string with variables.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>template</td>
<td>String</td>
<td>String template with values for substitution.</td>
</tr>
<tr>
<td>data</td>
<td>Object</td>
<td>Object containing variables for substitution.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A formatted string.</td>
</tr>
</tbody>
</table>

```javascript
spUtil.format('An error occurred: {error} when loading {widget}',
  {error: '404', widget: 'sp-widget'})
```

Output:

'An error occurred: 404 when loading sp-widget'

**spUtil - refresh(Object $scope)**

Calls the server and replaces the current **options** and **data** with the server response. Calling spUtil.refresh() is similar to calling server.refresh(). However, when you call spUtil.refresh(), you can define the $scope object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$scope</td>
<td>Object</td>
<td>The scope defined for the update.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The updated options and data objects.</td>
</tr>
</tbody>
</table>

**spUtil - update(Object $scope)**

Updates the data object on the server within a given scope.

This method is similar to `server.update()`, but includes a $scope parameter that defines the scope to pass over.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$scope</td>
<td>Object</td>
<td>The scope defined for the update.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The updated data object.</td>
</tr>
</tbody>
</table>

**spUtil - recordWatch(Object $scope, String table, String filter, Function callback)**

Watches for updates to a table or filter and returns the value from the callback function.

Allows a widget developer to respond to table updates in real-time. For instance, by using `recordWatch()`, the Simple List widget can listen for changes to its data table. If records are added, removed, or updated, the widget updates automatically.

When passing the $scope argument into the `recordWatch()` function, be sure to inject $scope into the parameters of your client script function.

Tables that are periodically subject to a high frequency of database events are blacklisted from `recordWatch()` to prevent event storms.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$scope</td>
<td>Object</td>
<td>Scope of the data object updated by the callback function.</td>
</tr>
<tr>
<td>table</td>
<td>String</td>
<td>Watched table.</td>
</tr>
<tr>
<td>filter</td>
<td>String</td>
<td>Filter for fields to watch.</td>
</tr>
<tr>
<td>callback</td>
<td>Function</td>
<td>Optional. Parameter to define the callback function.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Promise</td>
<td>Return value of the callback function.</td>
</tr>
</tbody>
</table>

//A simple recordWatch function.  
spUtil.recordWatch($scope, "live_profile", "sys_id=" + liveProfileId);

//In a widget client script  
function(spUtil, $scope) {  
  /* widget controller */  
  var c = this;

  // Registers a listener on the incident table with the filter  
  active=true,  
  // meaning that whenever something changes on that table with  
  that filter,  
  // the callback function is executed.  
  // The callback function takes a single parameter 'response',  
  // which contains  
  // the property 'data'. The 'data' property contains  
  // information about the changed record.  
  spUtil.recordWatch($scope, "incident", "active=true",  
  function(response) {  
    // Returns the data inserted or updated on the table  
    console.log(response.data);
  });
}

**StandardCredentialsProvider**

Use StandardCredentialsProvider API to retrieve credential information.

You can use this API in scoped applications, or within the global scope. In scoped scripts, use the sn_cc namespace identifier.

This function retrieves credential information by sys ID and by given credential attributes.

//Get a single credential

```javascript
var provider = new sn_cc.StandardCredentialsProvider();
var credential = provider.getCredentialByID("put a sys_id here");
var userName = credential.getAttribute("user_name");
var password = credential.getAttribute("password");
//using getAttribute for new keys in extended tables, for example  
//cloud management credential has the "user_public_key" attribute  
var userPublicKey = credential.getAttribute("user_public_key");
```

//Get a list of SSH credentials

```javascript
var provider = new sn_cc.StandardCredentialsProvider();
var credentials = provider.getCredentials(["ssh"]);
for (var i = 0; i < credentials.length; i++) {
```
var credential = credentials[i];
    gs.info(credential.getAttribute("name"));
}

**StandardCredentialsProvider - StandardCredentialsProvider()**

Use StandardCredentialsProvider() to retrieve credential information.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Scoped StandardCredentialsProvider - getCredentials(String)**

This function returns an array of all credentials that match the given types and tags.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>types</td>
<td>String</td>
<td>Types is an array of credential type names. For example, (&quot;ssh&quot;, &quot;windows&quot;)</td>
</tr>
<tr>
<td>handles</td>
<td>String</td>
<td>Handles is a comma-separated list of handle names. For example, &quot;ssh,jdbc&quot;</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StandardCredential</td>
<td>Information about the ConnInfo...</td>
</tr>
</tbody>
</table>

var provider = new sn_cc.StandardCredentialsProvider();
var credentials = provider.getCredentials(['ssh']);
for (var i = 0; i < credentials.length; i++) {
    var credential = credentials[i];
    gs.info(credential.getAttribute("name"));
}

**Scoped StandardCredentialsProvider - getCredentialByID(String)**

This function retrieves a credential object identified by the given sys ID.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys ID</td>
<td>String</td>
<td>A string representing the sys ID of the credential record.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StandardCredential</td>
<td>A credential.</td>
</tr>
</tbody>
</table>

```javascript
var provider = new sn_cc.StandardCredentialsProvider();
var credentials = provider.getCredentials(["ssh"]);
for (var i = 0; i < credentials.length; i++) {
    var credential = credentials[i];
    gs.info(credential.getAttribute("name"));
}
```

StartDiscovery

Starts discovery jobs.
Use this API during the discovery scanning phase.

StartDiscovery - addAdditionalIP(String status, String ip, String sensor)

Adds an IP (or comma-separated list of IPs) to an existing discovery (used by networks discovery).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>String</td>
<td>The discovery status</td>
</tr>
<tr>
<td>ip</td>
<td>String</td>
<td>An IP address, or a comma-separated list of IP addresses.</td>
</tr>
<tr>
<td>sensor</td>
<td>String</td>
<td>The discovery sensor</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

StartDiscovery - discoverFromSchedule(String ip, Boolean samePhase, String sensor)

Handles discovery of devices (whether basic or advanced) from a schedule.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>String</td>
<td>An IP address, or a comma-separated list of IP addresses.</td>
</tr>
<tr>
<td>samePhase</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>sensor</td>
<td>String</td>
<td>The discovery sensor</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**StartDiscovery - getCancelScript**

Returns the cancel script.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The cancel script</td>
</tr>
</tbody>
</table>

**StartDiscovery - scheduleCancelJob**

If the schedule specifies a cancel time, schedules a job to cancel if necessary.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**StartDiscovery - startFromIp**

Starts a discovery for a single IP using the specified schedule to get information such as MID servers and behaviors.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>schedule</td>
<td>DiscoverySchedule</td>
<td>A DiscoverySchedule instance for the schedule to use.</td>
</tr>
<tr>
<td>ip</td>
<td>String</td>
<td>The IP address to discover in a dot-formatted string.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the status record created.</td>
</tr>
</tbody>
</table>

#### StartDiscovery - startFromSchedule(GlideRecord schedule, GlideRecord job)

Invoked by the discovery script include to kick off a scheduled (or discover now) discovery.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>schedule</td>
<td>GlideRecord</td>
<td>The instance of discovery_schedule that this discovery is based on.</td>
</tr>
<tr>
<td>job</td>
<td>GlideRecord</td>
<td>The instance of sys_trigger that triggered this discovery.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### StateManagementScriptableApi

Provides a set of methods for manipulating CI operational states and applying CI actions.

The StateManagementScriptableApi methods adhere to restrictions and allowances specified by not allowed CI actions, compatible CI actions, and not allowed operational transitions. If a method attempts to perform a restricted operation, the operation is blocked, an error is logged, and a task is created if appropriate.

The StateManagementScriptableApi methods are static methods. You access the methods using the `SNC.StateManagementScriptableApi` global object.

#### StateManagementScriptableApi - addBulkCIAction(String requestorId, String sysIdList, String ciActionName, String ciActionListOld, String leaseTime)

Add a CI action to a list of CIs.
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestorId</td>
<td>String</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator() method.</td>
</tr>
<tr>
<td>sysIdList</td>
<td>String</td>
<td>A list comma separated list of CI sys_ids.</td>
</tr>
<tr>
<td>ciActionName</td>
<td>String</td>
<td>The CI action name.</td>
</tr>
<tr>
<td>ciActionListOld</td>
<td>String</td>
<td>(Optional) A comma separated list of old CI actions that all CIs should be in.</td>
</tr>
<tr>
<td>leaseTime</td>
<td>String</td>
<td>(Optional) Time duration for which the lease is valid for specified CI Action. In the format HH:MM:SS.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| String | A JSON formatted string with name-value pairs for  
|        | · result - true if the action is set for all CIs in the list; otherwise, false.  
|        | · errors - list of errors.                                                                                                                  |

```java
StateManagementScriptableApi - extendCIActionLease(String requestorId, String ciSysId, String ciActionName, String leaseTime)
```

Extend the CI-action-lease time for the registered user. If the previous lease has expired, the new lease time starts now.

## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestorId</td>
<td>String</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator() method.</td>
</tr>
<tr>
<td>ciSysId</td>
<td>String</td>
<td>The CI's sys_id.</td>
</tr>
<tr>
<td>ciActionName</td>
<td>String</td>
<td>The CI action name.</td>
</tr>
<tr>
<td>leaseTime</td>
<td>String</td>
<td>Time duration for which the lease is valid for specified CI Action. In the format HH:MM:SS.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>A JSON formatted string with name-value pairs for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>· result - true if the lease time has been set.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>· errors - list of errors.</td>
<td></td>
</tr>
</tbody>
</table>

**StateManagementScriptableApi - getCIActions(String ciSysId)**

Returns a list of active CI actions for the specified CI.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciSysId</td>
<td>String</td>
<td>The sys_id of the CI</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string with name-value pairs for</td>
</tr>
<tr>
<td></td>
<td>· ciActions - a comma separated list of active CI actions for the CI, or NO_ACTIVE_ACTION.</td>
</tr>
<tr>
<td></td>
<td>· errors - list of errors.</td>
</tr>
</tbody>
</table>

**StateManagementScriptableApi - getOperationalState(String ciSysId)**

Returns the CI's operational state.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciSysId</td>
<td>String</td>
<td>Sys_id of the CI</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string with name-value pairs for</td>
</tr>
<tr>
<td></td>
<td>· operationalStates - one of the string choice values of the operational_status field, or UNKNOWN.</td>
</tr>
<tr>
<td></td>
<td>· errors - list of errors.</td>
</tr>
</tbody>
</table>

**StateManagementScriptableApi - isCompatibleCIAction(String actionName, String otherActionName)**

Determines if the two specified actions are compatible.
StateManagementScriptableApi - isLeaseExpired(String requestorId, String ciSysId, String ciActionName)

Determines if the lease has expired for the requestor of a specified CI Action.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestorId</td>
<td>String</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator() method.</td>
</tr>
<tr>
<td>ciSysId</td>
<td>String</td>
<td>The CI's sys_id.</td>
</tr>
<tr>
<td>ciActionName</td>
<td>String</td>
<td>Name of the CI action.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string with name-value pairs for · result - true if the user's lease has expired. · errors - list of errors.</td>
</tr>
</tbody>
</table>

StateManagementScriptableApi - isNotAllowedAction(String ciType, String opsLabel, String actionName)

For a type of CI, determine if a CI action is not allowed for an operational state.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciType</td>
<td>String</td>
<td>The CI type</td>
</tr>
<tr>
<td>opsLabel</td>
<td>String</td>
<td>The operational state</td>
</tr>
<tr>
<td>actionName</td>
<td>String</td>
<td>The CI action name</td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if the action is not allowed on the specified CI type in the specified operational state.</td>
</tr>
</tbody>
</table>

### StateManagementScriptableApi - isNotAllowedOpsTransition(String ciType, String opsLabel, String transitionOpsLabel)

For a type of CI, determine if an operational state transition is not allowed.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciType</td>
<td>String</td>
<td>The CI type</td>
</tr>
<tr>
<td>opsLabel</td>
<td>String</td>
<td>The label of the beginning operational state.</td>
</tr>
<tr>
<td>transitionOpsLabel</td>
<td>String</td>
<td>The label of the ending operational state.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Returns true if specified operational state transition is not allowed on the specified CI type.</td>
</tr>
</tbody>
</table>

### StateManagementScriptableApi - isValidRequestor(String requestorId)

Determine if the specified requestor is a valid active workflow user or a registered user.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestorId</td>
<td>String</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator() method.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string with name-value pairs for</td>
</tr>
<tr>
<td></td>
<td>• result - true if the requestorId is valid.</td>
</tr>
<tr>
<td></td>
<td>• errors - list of errors.</td>
</tr>
</tbody>
</table>
**StateManagementScriptableApi - registerOperator()**

Register an operator for a non-workflow user.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| String | A JSON formatted string with name-value pairs for  
         - requestorId - Registered user GUID that is used to set CI Action/operational states.  
         - result - true if successfully registered.  
         - errors - list of errors. |

```javascript
var output = 
  SNC.StateManagementScriptableApi.registerOperator();
var jsonUntil = new JSON();
var result = jsonUntil.decode(output);
var requestorId = result.requestorId;
```

**StateManagementScriptableApi - removeBulkCIAction(String requestorId, String sysIdList, String ciActionName)**

Removes a CI action for a list of CIs.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestorId</td>
<td>String</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator() method.</td>
</tr>
<tr>
<td>sysIdList</td>
<td>String</td>
<td>A comma separated list of CI sys_ids</td>
</tr>
<tr>
<td>ciActionName</td>
<td>String</td>
<td>The CI action name</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string with name-value pairs for</td>
</tr>
<tr>
<td></td>
<td>· result - true if the action is removed for all CIs in the list; otherwise, false.</td>
</tr>
<tr>
<td></td>
<td>· errors - list of errors.</td>
</tr>
</tbody>
</table>

**StateManagementScriptableApi - setBulkClOperationalState(String requestorId, String sysIdList, String opsLabel, String opsStateListOld)**

Set the operational state for list of CIs.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestorId</td>
<td>String</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator() method.</td>
</tr>
<tr>
<td>sysIdList</td>
<td>String</td>
<td>A comma separated list of CI sys_ids.</td>
</tr>
<tr>
<td>opsLabel</td>
<td>String</td>
<td>This is the string label of an operational_status choice.</td>
</tr>
<tr>
<td>opsStateListOld</td>
<td>String</td>
<td>(Optional) A comma separated list of old CI states that all CIs should be in.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string with name-value pairs for</td>
</tr>
<tr>
<td></td>
<td>· result - true if the state is set for all CIs in the list; otherwise, false.</td>
</tr>
<tr>
<td></td>
<td>· errors - list of errors.</td>
</tr>
</tbody>
</table>

```javascript
var output = SNC.StateManagementScriptableApi.registerOperator();
var jsonUntil = new JSON();
var result = jsonUtil.decode(output);
var requestorId = result.requestorId;

// list of sys_ids to update
var sys_ids;

// Set list of sys_ids's Operational State to 'Repair in Progress'
```

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
output = 
SNC.StateManagementScriptableApi.setBulkCIOperationalState(requestorId, sys_ids,'Repair in Progress');
gs.print(output);

StateManagementScriptableApi - unregisterOperator( String requestorId)
Unregister an operator for non-workflow users.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestorId</td>
<td>String</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator() method.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A JSON formatted string with name-value pairs for</td>
</tr>
<tr>
<td></td>
<td>- result - true if successfully unregistered.</td>
</tr>
<tr>
<td></td>
<td>- errors - list of errors.</td>
</tr>
</tbody>
</table>

StopWatch
Use a StopWatch object to measure the duration of operations.
The StopWatch API can be used in client-side scripts using ListV2 and ListV3 APIs.

StopWatch - getTime()
Returns the number of milliseconds since the timer started.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of milliseconds since the timer started.</td>
</tr>
</tbody>
</table>

StopWatch - restart()
Resets the timer to the current time.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

StopWatch - StopWatch()

Creates an instance of the StopWatch class.
Uses the current time as the start time.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

StopWatch - StopWatch(Date initialDate)

Creates an instance of the StopWatch class using the specified date as the initial value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialDate</td>
<td>Date</td>
<td>The initial date for the object.</td>
</tr>
</tbody>
</table>

StopWatch - toString()

The elapsed time as HH:MM:SS.SSS.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The elapsed time formatted as HH:MM:SS.SSS.</td>
</tr>
</tbody>
</table>

SummaryTableWriter

The SummaryTableWriter script include enables the creation of summary sets for reports that can be generated with standard queries and trending.
This script provides methods that enable you to customize the output of the summary sets for reports, such as adding a title, defining a query on which to base the report data, and specifying the stack column and X axis.

**SummaryTableWriter - generate()**

Creates the summary set of records for reports.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var opened = new SummaryTableWriter('incident', '');
opened.generate();
```

**SummaryTableWriter - setQuery(String query)**

Defines the query to use to load the data for the summary record set.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>String</td>
<td>Query to use to load the summary data.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var summarySet = new SummaryTableWriter('incident', 'category');
summarySet.setQuery("active=true");
summarySet.generate();
```

**SummaryTableWriter - setStackOn(String column)**

Defines the column on which to stack the incident records.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>String</td>
<td>Column that is used to stack the incident records.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var summarySet = new SummaryTableWriter('incident', 'category');
summarySet.setQuery("active=true");
summarySet.setStackOn('priority');
summarySet.generate();
```

### SummaryTableWriter - setTitle(String title)

Sets the title of the summary set, which is also the chart title.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td>Title given to the summary set that is used as the chart title. Also used as the key field for chart generators to determine whether the data was generated.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var opened = new SummaryTableWriter('incident', '');
opened.setTitle("Trend of Open Incidents");
opened.setTrend('opened_at', 'month');
opened.generate();
```

### SummaryTableWriter - setTrend(String field, String interval)

Sets the time field used for the X axis or stacking field; can also set the time interval used to calculate the trend.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>String</td>
<td>Time field that is used for X axis values if grouping is not used. If grouping is used, then this is the stacking field.</td>
</tr>
<tr>
<td>interval</td>
<td>String</td>
<td>Time interval that is used to calculate the trend.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var opened = new SummaryTableWriter('incident', '');
opened.setTitle("Trend of Open Incidents");
opened.setTrend('opened_at', 'month');
opened.generate();
```

### SummaryTableWriter - SummaryTableWriter(String tableName, String columnName)

Produces a summary set for the COUNT of all incidents grouped by category.

The custom chart definition table contains a field named **Table** that is hidden by default. If the field is set, which it is in some of the base custom charts, then, no matter what is defined within the custom chart scripts, clicking on the chart points you to this table.

For example, if you create a custom chart against the `[task]` table through a script, but you copied the custom chart from a custom chart that contains `Table = incident`, then clicking on any bar in the custom script will go to incident.do rather than to task.do.

To check the value of the **Table** field, click the additional actions icon and select **Configure > Form Layout** and move the **Table** field from **Available** to **Selected**. Open the custom chart definition and set the **Table** field to **none**.

```javascript
var summarySet = new SummaryTableWriter('incident', 'category');
summarySet.generate();
```
TableUtils
TableUtils is a class of shortcuts for accessing table related information.
The TableUtils class is available to server-side scripts.

TableUtils - tableExists()
Checks to see if a table exists.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the table exists.</td>
</tr>
</tbody>
</table>

```javascript
var table = new TableUtils("my_table");
gs.print("Does 'my_table' exist? " + table.tableExists());
```

Output: Does ‘my_table’ exist? false

TableUtils - drop(String tableName)
Drops a database table.

**Note:** Use with extreme caution. Dropping a table permanently deletes the table and all its data. If the table is extended, use dropTableAndExtensions.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>Name of the table to drop</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var tu = new TableUtils();
tu.drop("table_that_will_be_lost_forever");
```
Output:

```javascript
var tu = new TableUtils();
tu.dropAndClean("table_that_will_be_lost_forever");
```

Output:

```javascript
dropping table table_that_will_be_lost_forever
Starting cache flush
Cache flush complete
TABLE DROP: admin dropped table table_that_will_be_lost_forever
*** Script: removing gauges for table_that_will_be_lost_forever
*** Script: removing forms for table_that_will_be_lost_forever
*** Script: removing styles for table_that_will_be_lost_forever
*** Script: removing forms sections for table_that_will_be_lost_forever
*** Script: removing lists for table_that_will_be_lost_forever
*** Script: removing related lists for table_that_will_be_lost_forever
*** Script: removing references to table_that_will_be_lost_forever
*** Script: removing dictionary entries for table_that_will_be_lost_forever
Background message, type:info, message: Table deleted
```
Note: Use with extreme caution. Dropping a database table permanently deletes the table and all of its data.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table to drop</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var tu = new TableUtils();
tu.dropAndClean("table_that_will_be_lost_forever");
```

Output:

dropping table parent_table_that_will_be_lost_forever
Starting cache flush
Cache flush complete
TABLE DROP: admin dropped table
  ext_table_that_will_be_lost_forever
  removing gauges for ext_table_that_will_be_lost_forever
  removing forms for ext_table_that_will_be_lost_forever
  removing styles for ext_table_that_will_be_lost_forever
  removing forms sections for ext_table_that_will_be_lost_forever
  removing lists for ext_table_that_will_be_lost_forever
  removing related lists for ext_table_that_will_be_lost_forever
  removing references to ext_table_that_will_be_lost_forever
  removing dictionary entries for
  ext_table_that_will_be_lost_forever
Background message, type:info, message: Table deleted
dropping table parent_table_that_will_be_lost_forever
Starting cache flush
Cache flush complete
TABLE DROP: admin dropped table
  parent_table_that_will_be_lost_forever
  removing gauges for parent_table_that_will_be_lost_forever
  removing forms for parent_table_that_will_be_lost_forever
  removing styles for parent_table_that_will_be_lost_forever
  removing forms sections for
  parent_table_that_will_be_lost_forever
  removing lists for parent_table_that_will_be_lost_forever
  removing related lists for
  parent_table_that_will_be_lost_forever
  removing references to parent_table_that_will_be_lost_forever
  removing dictionary entries for
  parent_table_that_will_be_lost_forever
Background message, type:info, message: Table deleted
TableUtils - getAbsoluteBase()

Returns the base table name from which the table was extended.

**Note:** For any table under the cmdb_ci hierarchy, this method returns cmdb_ci and not cmdb, which is the actual base table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Parameters**

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Base table name</td>
</tr>
</tbody>
</table>

```javascript
var table = new TableUtils("cmdb_ci_server");
gs.print(table.getAbsoluteBase());
```

Output: cmdb_ci

TableUtils - getAllExtensions()

Returns the list of tables that extend a table, includes the base table.

**Parameters**

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>A list of tables that extend the table, includes the base table.</td>
</tr>
</tbody>
</table>

```javascript
var table = new TableUtils("task");
gs.print(table.getAllExtensions());
```

Output:

`[task, incident, issue, kb_submission, sysapproval_group, change_request, change_request_imac, sc_task]`
TableUtils - getHierarchy()

Returns a list of all classes participating in the hierarchy of the specified table.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>A list of all classes in the specified table's hierarchy.</td>
</tr>
</tbody>
</table>

```javascript
var table = new TableUtils("cmdb_ci_server");
gs.print(table.getHierarchy());
```


TableUtils - getTables()

Returns the table hierarchy.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>A list of table names in the parent hierarchy.</td>
</tr>
</tbody>
</table>

```javascript
var table = new TableUtils("incident");
gs.print(table.getTables());
```

Output: (incident, task)
TableUtils - `getTableExtensions()`

Returns a list of tables that extend a table.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>A list of table names that extend the table.</td>
</tr>
</tbody>
</table>

```javascript
var table = new TableUtils("task");
gs.print(table.getTableExtensions());
```

Output:

```javascript
[incident, issue, kb_submission, sysapproval_group, change_request, change_request_imac, sc_task, problem, sc_req_item, ticket, ast_transfer_order, planned_task, change_task, change_phase, sc_request]
```

TableUtils - `hasExtensions()`

Determines if a table has been extended.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the table has extensions.</td>
</tr>
</tbody>
</table>

```javascript
var table = new TableUtils("cmdb_ci_server");
gs.print(table.hasExtensions());
```

Output: true

TableUtils - `isBaseClass()`

Determines if a table is a base class, meaning it has no parents and has extensions.
For example, Task is a base class since it is not extended from another table and has tables extended from it. Sys_user is not a base class because it has no parents, but does not have extensions.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Boolean  | Flag that indicates whether a table is a base class table meaning it has no parents but has extensions. Valid values:  
  - true: Table is a base class.  
  - false: Table is not a base class. |

### Example

```javascript
var table = new TableUtils("task");
gs.print("Task is base class: " + table.isBaseClass());

var table = new TableUtils("sys_user");
gs.print("User is base class: " + table.isBaseClass());
```

Output:

```
Task is base class: true
User is base class: false
```

### TableUtils - isSoloClass()

Determines if the table has no parents and no extensions.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the table has no parent and no table is extended from it.</td>
</tr>
</tbody>
</table>
```
var table = new TableUtils("task");
gs.print("task is solo class: " + table.isSoloClass());

var table = new TableUtils("cmdb_ci_win_server");
gs.print("cmdb_ci_win_server is solo class: " + table.isSoloClass());

var table = new TableUtils("sys_user");
gs.print("sys_user is solo class: " + table.isSoloClass());
```

Output:
```
task is solo class: false
cmdb_ci_win_server is solo class: false
sys_user is solo class: true
```

### TableUtils - TableUtils(String tableName)

Creates an instance of a TableUtils class.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>String</td>
<td>The table name</td>
</tr>
</tbody>
</table>

```
var tu = new TableUtils("incident");
```

### TemplatePrinter

Scoped TemplatePrinter handles printing from a mail script to the email message.

There is no constructor for the scoped TemplatePrinter API. The methods are called in mail scripts using the `template` global variable.

### Scoped TemplatePrinter - print(String string)

Prints the string to the email body.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>String</td>
<td>The string to print</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
template.print("Incident number - " + current.number + "\n");

**Scoped TemplatePrinter - space(Number spaces)**

Adds non-breaking spaces to the email body.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spaces</td>
<td>Number</td>
<td>The number of non-breaking spaces to output to the email body.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

template.space(4);

**TimelineSpan**

This class defines a set of properties that describe the characteristics and interactive behavior of an element rendered within a `TimelineItem`.

Since it is extremely important for all of a `TimelineItem`'s collection of spans to be unique, the creation of a new instance should be performed via the `createTimelineItem` method of an existing `TimelineItem` instance.

**TimelineSpan - addPredecessor(Object Array objArray)**

Adds multiple relationships between the current instance and other `TimelineSpan` objects by enumerating through the array of JavaScript objects.

Each object should have an internal property `relationship_sys_id` and `predecessor_sys_id` specified.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objArray</td>
<td>Object Array</td>
<td>JavaScript object array that contains two internal properties: <code>relationship_sys_id</code> and <code>predecessor_sys_id</code>.</td>
</tr>
</tbody>
</table>
**TimelineSpan - addPredecessor(String strPredecessorSysId, String strRelationshipSysId, String strTableName)**

Adds the specified relationship between the current instance and another TimelineSpan and allows the relationship to open a GlideWindow to display information about the relationship.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strPredecessorSysId</td>
<td>String</td>
<td>The sys ID of the planned task that is the predecessor of the relationship.</td>
</tr>
<tr>
<td>strRelationshipSysId</td>
<td>String</td>
<td>The sys ID of the relationship of the relationship.</td>
</tr>
<tr>
<td>strTableName</td>
<td>String</td>
<td>The name of the table for the relationship.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**TimelineSpan - addPredecessor(String strPredecessorSysId, String strRelationshipSysId)**

Adds the specified relationship between the current instance and another TimelineSpan with sys ID strPredecessorSysId.

The drawn line will not have any double click handlers associated with it.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strPredecessorSysId</td>
<td>String</td>
<td>The sys ID of the planned task that is the predecessor of the relationship.</td>
</tr>
<tr>
<td>strRelationshipSysId</td>
<td>String</td>
<td>The sys ID of the relationship of the relationship.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
**TimelineSpan - getAllowXDragLeft()**

Returns the boolean value of the `AllowXDragLeft` property.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the object’s start time can be adjusted; False otherwise.</td>
</tr>
</tbody>
</table>

**TimelineSpan - getAllowXDragRight()**

Returns the boolean value of the `AllowXDragRight` property.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the object’s end time can be adjusted; false otherwise.</td>
</tr>
</tbody>
</table>

**TimelineSpan - getAllowXMove()**

Returns the boolean value of the `AllowXMove` property.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the object can be moved; false otherwise.</td>
</tr>
</tbody>
</table>

**TimelineSpan - getAllowYMove()**

Returns the boolean value of the `AllowYMove` property.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the object can be moved vertically; false otherwise.</td>
</tr>
</tbody>
</table>

`TimelineSpan - getAllowYMovePredecessor()`

Returns the boolean value of the `AllowYMovePredecessor` property.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if a dashed relationship line can be drawn from the current object to a new successor; false otherwise.</td>
</tr>
</tbody>
</table>

`TimelineSpan - getInnerSegmentClass()`

Returns the name of the current inner segment class for the `TimelineSpan`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The name of the class for the current inner segment style.</td>
</tr>
</tbody>
</table>

`TimelineSpan - getInnerSegmentEndTimeMs()`

Returns the time in milliseconds of the end time of the inner segment portion of the `TimelineSpan`. 
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The end time of the TimelineSpan inner segment portion in milliseconds.</td>
</tr>
</tbody>
</table>

**TimelineSpan - getInnerSegmentStartDateTimeMs()**

Returns the time in milliseconds of the start time of the inner segment portion of the TimelineSpan.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The start time of the TimelineSpan inner segment portion in milliseconds.</td>
</tr>
</tbody>
</table>

**TimelineSpan - getIsChanged()**

Returns a boolean that specifies whether or not the current timeline item has been modified after initialization.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the current span has been marked as changed; otherwise false.</td>
</tr>
</tbody>
</table>

**TimelineSpan - getPointIconClass()**

Returns a String that specifies the name of the icon class to use for displaying the element on the timeline if the current instance has zero duration.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The name of the icon class to use for displaying the current TimelineSpan if the duration is zero.</td>
</tr>
</tbody>
</table>

### TimelineSpan - getPredecessors()

Returns an array of all the predecessor objects associated with the current instance. Each array object is a HashMap that contains a *predecessor_sys_id* and *relationship_sys_id* property.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Array</td>
<td>List of HashMaps that contain two internal properties: predecessor_sys_id and relationship_sys_id.</td>
</tr>
</tbody>
</table>

### TimelineSpan - getSpanColor()

Returns the string name of the color specified for displaying this span.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The HTML color name to use as the background color for the element.</td>
</tr>
</tbody>
</table>

### TimelineSpan - getSpanText()

Returns the string that specifies the text to display adjacent to the time element.
Note: This text only appears if the GlideTimeline object has enabled timeline text via glideTimeline.showTimelineText(true).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The text displayed adjacent to the element.</td>
</tr>
</tbody>
</table>

TimelineSpan - getStartTimeMs()

Returns the start time in milliseconds of the current TimelineSpan object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The start time of the element in milliseconds.</td>
</tr>
</tbody>
</table>

TimelineSpan - getSysId()

Returns the sys ID of the current object.

This method is useful for returning the sys Id when the current object instance was created without a specific sys Id to obtain the dynamically generated GUID.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The unique system ID of the current element.</td>
</tr>
</tbody>
</table>

TimelineSpan - getTable()

Returns the name of the table where the sys ID is referenced.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The table name.</td>
</tr>
</tbody>
</table>

**TimelineSpan - getTooltip()**

Returns the text/html to display in the tooltip when the `TimelineSpan` element is being hovered over.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The tooltip text.</td>
</tr>
</tbody>
</table>

**TimelineSpan - setAllowXDragLeft(Boolean bFlag)**

Sets a flag that determines whether the element’s start date can be dragged left or right therefore adjusting the duration of the task.

The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is `false`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bFlag</td>
<td>Boolean</td>
<td><code>true</code> to enable the element’s start date to be adjusted; <code>false</code> otherwise.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is **false**.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bFlag</td>
<td>Boolean</td>
<td><strong>True</strong> to enable the element's end date to be adjusted; <strong>false</strong> otherwise.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**TimelineSpan - setAllowXMove(Boolean bFlag)**

Sets a flag that determines whether the element can be moved to start at a different time. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is **false**.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bFlag</td>
<td>Boolean</td>
<td><strong>True</strong> to enable the element to be moved horizontally; <strong>false</strong> otherwise.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**TimelineSpan - setAllowYMove(Boolean bFlag)**

Sets a flag that determines whether the element can be dragged vertically on the timeline. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is **false**.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bFlag</td>
<td>Boolean</td>
<td><strong>True</strong> to enable the element to be moved vertically; <strong>false</strong> otherwise.</td>
</tr>
</tbody>
</table>
### TimelineSpan - `setAllowYMovePredecessor(Boolean bFlag)`

Sets a flag that determines whether a dashed relationship line can be drawn from this element interactively on the timeline.

The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is false.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bFlag</td>
<td>Boolean</td>
<td>True to enable a relationship line to be drawn from this element; false otherwise.</td>
</tr>
</tbody>
</table>

### TimelineSpan - `setInnerSegmentClass(String styleClass)`

Specifies the name of the class to use for stylizing the inner segment if it exists. The default value is `green`.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>styleClass</td>
<td>String</td>
<td>One of the following values: green, blue, or silver.</td>
</tr>
</tbody>
</table>

### TimelineSpan - `setInnerSegmentTimeSpan(Number startTimeMs, Number endTimeMs)`

Creates an inner segment to show within the current timespan defined by the range specified.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTimeMs</td>
<td>Number</td>
<td>The start time in milliseconds.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>endTimeMs</td>
<td>Number</td>
<td>The end time in milliseconds.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**TimelineSpan - setPointIconClass(String iconClassName)**

Sets the icon class to use for displaying the current element on the timeline if the current instance has zero duration.

**Note:** This only affects the current TimelineSpan object and will take precedence over the defaultPointIconClass specified by the GlideTimeline.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iconClassName</td>
<td>String</td>
<td>String that specifies one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- milestone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- blue_square</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- sepia_square</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- green_square</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- red_square</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- black_square</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- blue_circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- sepia_circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- green_circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- red_circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- black_circle</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**TimelineSpan - setSpanColor(String strColor)**

Sets the color for displaying this span.
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strColor</td>
<td>String</td>
<td>The HTML color name for the color of this span.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### TimelineSpan - setSpanText(String strSpanText)

Sets the text to display adjacent to the time element.

**Note:** This text will only be displayed if the GlideTimeline object has enabled timeline text via `glideTimeline.showTimelineText(true)`.

## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strSpanText</td>
<td>String</td>
<td>The text to display next to the time element.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### TimelineSpan - setTimeSpan(Number nStartTime, Number nEndTimeMs)

Sets the start and end dates for the current span.

## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nStartTime</td>
<td>Number</td>
<td>The start time in milliseconds.</td>
</tr>
<tr>
<td>nEndTimeMs</td>
<td>Number</td>
<td>The end time in milliseconds.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### TimelineSpan - setTimeSpan(String strStartTime, String strEndTimeMs)

Sets the start and end times for the current span.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strStartTime</td>
<td>String</td>
<td>The start time in milliseconds.</td>
</tr>
<tr>
<td>strEndTimeMs</td>
<td>String</td>
<td>The end time in milliseconds.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

TimelineSpan - setTooltip(String strTooltipText)

Sets the text to display in the tooltip when the TimelinSpan element is being hovered over.

**Note:** You can specify valid HTML in the string that sets the tooltip.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strTooltipText</td>
<td>String</td>
<td>The text to display in the tooltip.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

TaskStateUtil

The TaskStateUtil API is in the task state management utility script include and is used for working with task-type table state attributes.

The TaskStateUtil API is primarily used by the Task Active State Management business rule to set the active field based on state changes. Configurations are defined in the task.state dictionary element, usually using dictionary overrides since state values vary by table.

The TaskStateUtil API can be called by any server script to determine inactive states, default work, or default close states for a table.

The required attributes are defined on the planned_task table so all planned task types are supported. We will eventually add the attributes to other task types and eventually the base task table. You are free to do this if you want to leverage this feature now.

These attributes can be defined on the task.state dictionary element or a dictionary override for extended task tables.
### Related Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>close_states</td>
<td>Semicolon delimited list of state values that are inactive, used to identify whether the task should be set to active or inactive. This is a required attribute to use the TaskStateUtil functionality.</td>
</tr>
<tr>
<td>default_close_state</td>
<td>Optional attribute to define the state value of the default close state if you want to define business rules that automatically close a task. Defaults to 3, typically Closed Complete if attribute is not defined.</td>
</tr>
<tr>
<td>default_work_state</td>
<td>Optional attribute to define the state value of the default working state if you want to define business rules that automatically set a task for working. Defaults to 2, typically Work in Progress if the attribute is not defined.</td>
</tr>
</tbody>
</table>

**TaskStateUtil - ATTR_DEFAULT_WORK**

The name of the attribute that identifies default work state.

**Field**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTR_DEFAULT_WORK</td>
<td>String</td>
<td>Identifies default work state. Value: default_work_state</td>
</tr>
</tbody>
</table>

**TaskStateUtil - ATTR_DEFAULT_CLOSE**

The name of the attribute that identifies the default close state.

**Field**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTR_DEFAULT_CLOSE</td>
<td>String</td>
<td>Identifies the default close state. Value: default_close_state</td>
</tr>
</tbody>
</table>

**TaskStateUtil - ATTR_INACTIVE_STATES**

The name of the attribute that identifies inactive states.

**Field**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTR_INACTIVE_STATES</td>
<td>String</td>
<td>Identifies inactive states. Value: close_states</td>
</tr>
</tbody>
</table>

**TaskStateUtil - getDefaultCloseState**

Returns the value for the default closed state.

The default closed state value is 3 if the default_close_state attribute has not been specified.
### TaskStateUtil - getDefaultCloseState

Returns the value for the default close state.

The default close state value is 2 if the default_work_state attribute has not been specified.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var stateUtil = new TaskStateUtil(current);
//get the close state
var defaultCloseState = stateUtil.getDefaultCloseState();
current.state = defaultCloseState;
```

### TaskStateUtil - getDefaultWorkState

Returns the value for the default work state.

The default work state value is 2 if the default_work_state attribute has not been specified.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var stateUtil = new TaskStateUtil(current);
//get the work state
var defaultWorkState = stateUtil.getDefaultWorkState();
current.state = defaultWorkState;
```

### TaskStateUtil - getInactiveStates

Returns a list of the inactive state values.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
```
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>Array of state values that are inactive.</td>
</tr>
</tbody>
</table>

```javascript
var stateUtil = new TaskStateUtil(current);
//get the inactive state values
var inactiveStates = stateUtil.getInactiveStates();
```

**TaskStateUtil - isStateInactive(String state)**

Returns the active status of the specified state.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>String</td>
<td>The state value to check.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the state is inactive.</td>
</tr>
</tbody>
</table>

```javascript
var stateUtil = new TaskStateUtil(current);
var previousStateInactive =
    stateUtil.isStateInactive(previous.state);
var currentStateInactive =
    stateUtil.isStateInactive(current.state);
```

**TaskStateUtil - runMarkClosed**

Decides whether the **mark closed** business rule should be run or not.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Whether the business rule should be allowed to run or not.</td>
</tr>
</tbody>
</table>
TaskStateUtil - runTaskCloser
Decides whether the task closer business rule should be run or not.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Determines whether the business rule should be allowed to run or not.</td>
</tr>
</tbody>
</table>

TaskStateUtil - runTaskReopener
Decides whether the task reopener business rule should be run or not.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Whether the business rule should be allowed to run or not.</td>
</tr>
</tbody>
</table>

TaskStateUtil - setDefaultWorkState(String defaultWorkState)
Enables the user to specify their own default work state.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaultWorkState</td>
<td>String</td>
<td>The value to use for the default work state.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaskStateUtil</td>
<td>A self-reference to allow for method chaining.</td>
</tr>
</tbody>
</table>

TaskStateUtil - SYSTEM_DEFAULT_CLOSE
The value of the default close state is Closed Complete on the Task table.
<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SYSTEM_DEFAULT_CLOSE</td>
<td>Integer</td>
<td>Value of the default close state is Closed Complete on the Task table. Value: 3</td>
</tr>
</tbody>
</table>

TaskStateUtil - SYSTEM_DEFAULT_WORK

The value of the default work state is Work in progress on the Task table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SYSTEM_DEFAULT_WORK</td>
<td>Integer</td>
<td>Value of the default work state is Work in progress on the Task table. Value: 2</td>
</tr>
</tbody>
</table>

TaskStateUtil - SYSTEM_INACTIVE_STATES

The values of the default inactive states: Closed Complete, Closed Incomplete, Closed Skipped on the Task table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SYSTEM_INACTIVE_STATES</td>
<td>Integer array</td>
<td>Values of the default inactive states: Closed Complete, Closed Incomplete, Closed Skipped on the Task table. Value: 3, 4, 7</td>
</tr>
</tbody>
</table>

TaskStateUtil - TaskStateUtil(GlideRecord task)

Creates a TaskStateUtil object.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>task</td>
<td>GlideRecord</td>
<td>This must be a GlideRecord from a task table.</td>
</tr>
</tbody>
</table>

```
var stateUtil = new TaskStateUtil(current);
```

Transformer

Manipulate time-series data to prepare the data for evaluation and analysis.

The Transformer class can be used in scoped and global server scripts. When using the Transformer class, use the sn_clotho namespace identifier.
The general use case is to determine the period to be evaluated, select the records from the table with the metric field, define the type of transform to run, and then execute the transform.

```javascript
// create the start and end time
var start = new GlideDateTime();
start.addSeconds(-1 * 60 * 60);
var end = new GlideDateTime();

//mb_demo_drone is a table with metric fields.
var drones = new GlideRecord("mb_demo_drone");
drones.addQuery("model", "Kingfisher Phantom");
drones.query();

//build a transform that returns a simple average
var builder = new sn_clotho.Transformer(drones);
builder.metric("mb_demo_mt_rem_battery").avg().label("Original");

//execute transform and return result for visualization
var result = builder.execute(start,end);
```

This class is part of the MetricBase application.

**Scoped Transformer - execute(GlideDateTime start, GlideDateTime end)**

Run the transform.

Use the `metric()` and `groupBy()` methods before calling `execute()`. The `execute()` method can only be called once for each Transformer object.

Actions performed as part of the transform do not change the data in the MetricBase database.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>GlideDateTime</td>
<td>The beginning of the period to be evaluated.</td>
</tr>
<tr>
<td>end</td>
<td>GlideDateTime</td>
<td>The end of the period to be evaluated.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformResult</td>
<td>The transformed data.</td>
</tr>
</tbody>
</table>

```javascript
var minutesAgoStart = 60;
var end = new GlideDateTime();
var start = new GlideDateTime(end);
start.addSeconds(-1 * 60 * minutesAgoStart);

// query subject records
var grDrone = new GlideRecord('mb_demo_drone');
grDrone.query();

// building transform; get the average transforms of a metric, grouping by model
var transformer = new sn_clotho.Transformer(grDrone);
```
transformer.groupBy("fleet").metric("mb_demo_mt_altitude").avg().label('avg - %g:fleet:');

// execute and return result for visualization
var tfrmResult = transformer.execute(start, end);

**Scoped Transformer - groupBy(String field)**

Specify a field to be used to group the data.

If you are going to use the `groupBy()` method, it must be called before the `execute()` method.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>String</td>
<td>A field in the table to be used to group the transform results.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify the transform characteristics.</td>
</tr>
</tbody>
</table>

```javascript
var transformer = new sn_clotho.Transformer(grDrone);
var trnsfrm = transformer.groupBy("fleet");
```

**Scoped Transformer - metric(String metricName)**

Specify the metric field to be used in the transform.

You can specify multiple metrics to be used in the transform. The `metric()` method cannot be called after the `execute()` method is called.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metricName</td>
<td>String</td>
<td>Name of the metric field.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify the transform characteristics.</td>
</tr>
</tbody>
</table>

```javascript
var transformer = new sn_clotho.Transformer(grDrone);
```


```javascript
var trnsfrm = transformer.metric("mb_demo_mt_altitude");
```

**Scoped Transformer - Transformer( GlideRecord sourceRecords)**

Create a Transformer object.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceRecords</td>
<td>GlideRecord</td>
<td>Contains the records for which metrics are to be evaluated. Can be one record or many.</td>
</tr>
</tbody>
</table>

```
//where drones is a GlideRecord created from a table with a metric field
var builder = new sn_clotho.Transformer(drones);
```

**TransformPart**

Use the TransformPart class to specify details of the transform to be done.

The TransformPart class can be used in scoped and global server scripts. When using the TransformPart class, use the `sn_clotho` namespace identifier.

There is no constructor for this class. TransformPart objects are returned by many Transformer and TransformPart methods.

The methods of this class define the transforms to be done. The actual transformation is done when the `execute()` method is called on the Transformer object.

The order the TransformPart methods are called is important.

- The `metric()` method must be called before calling a transform method.
- You cannot use the `metric()` or `groupBy()` methods after calling a transform method.
- Intermediate transforms are not returned in a result unless the `collect()` method is called for the intermediate result you want.

```
// where tp is a TransformPart object
// example 1
tp.avg().add(2);
var tr = tp.execute();
// tr contains avg+2, but not avg

// example 2
tp.avg().add(2);
tp.avg();
var tr = tp.execute();
// tr contains both avg and avg + 2

// example 3
tp.avg().collect().add(2);
var tr = tp.execute();
// tr contains both avg and avg + 2
```

This class is part of the MetricBase application.
**TransformPart - add(Number constant)**

Add the specified number to the value in each time stamp.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>Number</td>
<td>The number to add to the value in each time stamp.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

**TransformPart - avg()**

Aggregate the selected metric series into one series containing the average value for each time stamp.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

**TransformPart - bottom(Number count)**

Create a result set that for each time stamp returns specified number of bottom values. This method results in `count` number of series. Each value retains the label of its source series.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Number</td>
<td>The number of series to return. The series are labeled 0 to count - 1.</td>
</tr>
</tbody>
</table>
## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

### TransformPart - ceil(Number ceiling)

Replace the value in any time stamp that is greater than the specified value with the specified value.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ceiling</td>
<td>Number</td>
<td>The maximum allowed value for any time stamp.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

### TransformPart - collect()

Mark this transform for collection.

Transforms that are part of a chain, but not the last transform, are by default not collected. A collected transform is returned as part of the transform result.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

### TransformPart - count()

Aggregate the selected metric series into one series containing the number of values for each time stamp.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

#### TransformPart - div(Number constant)

Divide the value in each time stamp by the specified number.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>Number</td>
<td>The number by which to divide the value of each time stamp.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

#### TransformPart - filter(Object aggregator, Object duration)

Create a series using the specified aggregator for the specified time.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregator</td>
<td>Object</td>
<td>Can be AVG, MIN, MAX, or LAST</td>
</tr>
<tr>
<td>duration</td>
<td>Object</td>
<td>The time period for doing</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

#### TransformPart - floor(Number floor)

Replace the value in any time stamp that is less than the specified value with the specified value.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>floor</td>
<td>Number</td>
<td>The minimum value for any time stamp.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

**TransformPart - fractiles(Array fractions)**

Create series made up of the value that the specified percentage of values is below. Returns a series for each fraction in the specified array.

The value in a time stamp in a returned series is the value at which the specified fraction of the samples for that time stamp is below. For example, if the fraction is 0.5, then the value in the time stamp is the value where half the values in the input series are below (median).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fractions</td>
<td>Array of numbers</td>
<td>The fractions to use on the input series.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics. Contains one series for each fraction specified.</td>
</tr>
</tbody>
</table>

// returns a single series containing the median for each time stamp, which  
// means that half a time stamp's values are below the returned value  
fractiles([.5])  
// returns four series, one series for each of the 25%, 50%, 75%, and 100% quartiles  
fractiles([.25, .5, .75, 1])  
// returns the median, 95% percentile, the max value  
fractiles([.50, .95, 1])

**TransformPart - getResult()**

Return the part of the result relevant to this transform.
The `collect()` method must be called before the `execute()` method, and the `execute()` method must be called before calling the `getResult()` method.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformResult</td>
<td>Contains the transform results associated with this part of the transform.</td>
</tr>
</tbody>
</table>

```javascript
var t = new sn_clotho.Transformer(drones);
t.metric("mb_demo_mt_altitude");
var avgTform = t.avg();
t.execute();
var avgTformResult = avgTform.getResult();
```

**TransformPart - `groupBy(String field)`**

Specify a field to be used to group the data.

The `groupBy()` method cannot be called after a transform has been run.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>String</td>
<td>A field in the table to be used to group the transform results.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

**TransformPart - `interpolate(Object count)`**

Create a data value for a NaN data item by interpolating from adjacent data values.
**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Object</td>
<td>Specifies the number of data samples in each direction to check for a non NaN value. If a non NaN value is not found, NaN is used.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

**TransformPart - label(String label)**

Add a label for the resulting series.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>String</td>
<td>The label for the transform results.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

**TransformPart - limit(Object count)**

Returns at most the specified number of values, starting at the most recent non-NaN value.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Object</td>
<td>A number of time stamps.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>
TransformPart - log(Number base)

Run a logarithm on the value in each time stamp where the result is the log of the specified base for the time stamp value.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>Number</td>
<td>The base for the logarithm calculation.</td>
</tr>
</tbody>
</table>

TransformPart - max()

Returns a series with the maximum value for each time stamp.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

TransformPart - median()

Create a series containing the median of values for each time stamp across a set of series.

If there are n series in the TransformPart object, then if n is odd, the \((n / 2 + 1)\) value for a time stamp is the median. If n is even, the average of the \((n / 2)\) and \((n / 2 + 1)\) values for a time stamp is the median.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

**TransformPart - metric(String metric)**

Specify the metric field to be used in the transform.
You can specify multiple metrics to be used in the transform. The `metric()` method cannot be called after the transform has been run.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric</td>
<td>String</td>
<td>Name of the metric field.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

**TransformPart - min()**

Returns a series with the minimum value for each time stamp.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

**TransformPart - mul(Number constant)**

Multiply the value in each time stamp by the specified number.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>Number</td>
<td>The number by which to multiply the value of each time stamp.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

**TransformPart - partition(String aggregator, GlideDateTime duration, GlideDateTime base)**

Partition the series into intervals of the same duration.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregator</td>
<td>String</td>
<td>The aggregator to use. Can be min, max, avg, or last.</td>
</tr>
<tr>
<td>duration</td>
<td>GlideDateTime or an ISO 8601 formatted string</td>
<td>The interval length.</td>
</tr>
<tr>
<td>base</td>
<td>GlideDateTime or an ISO 8601 formatted string</td>
<td>The zero offset for partitioning. For example, if you partition by day (24h), then set the base to Monday at midnight in your time zone. If you partition by 30 days, then set the base to 1st day of the most recent month.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

**TransformPart - resample(Number count)**

Specify the number of samples to include in the result.

You can use the resample() method to reduce the number of samples in the result to more closely match the number of samples you are going to display.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Number</td>
<td>The number of samples to include in the result.</td>
</tr>
</tbody>
</table>
### TransformPart - round(Number precision)

Round the value in each time stamp to the specified precision.
Perform this calculation on each value.

\[(v / \text{precision}) \times \text{precision}\]

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>precision</td>
<td>Number</td>
<td>The value to be used in the rounding calculation.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

### TransformPart - stddev()

Create a series containing the standard deviation of values for each time stamp across a set of series.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

### TransformPart - sub(Object constant)

Subtract the specified number from the value in each time stamp.
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>Object</td>
<td>The number to subtract from the value in each time stamp.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

### TransformPart - sum()

Aggregate the selected metric series into one series containing the sum of all values for each time stamp.

## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>

### TransformPart - top(Number count)

Create a result set that for each time stamp returns the specified number of top values. This method results in `count` number of series. Each value retains the label of its source series.

## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Number</td>
<td>The number of series to return. The series are labeled 0 to count - 1.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformPart</td>
<td>A TransformPart object that can be used to specify transform characteristics.</td>
</tr>
</tbody>
</table>
**TransformResult**

Provides the result of a transformation run on time-series data.

The TransformResult class can be used in scoped and global server scripts. When using the Transformer class, use the sn_clotho namespace identifier.

There is no constructor for this class. TransformResult objects are returned by many TransformPart methods.

This class is part of the MetricBase application.

**TransformResult - byGroup()**

Returns an array of Data objects. Returns an error if no group was specified for the transform.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Array</td>
</tr>
</tbody>
</table>

**TransformResult - getByLabel(String label)**

Returns the transformed data with the specified label.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>label</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Data</td>
</tr>
</tbody>
</table>

**TransformResult - getData()**

Returns a single Data object, or null if the result is empty.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>
TransformResult - toArray()

Returns the transformed data as an array. This method turns a Data object into an array.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>The Data object formatted as an array.</td>
</tr>
</tbody>
</table>

UPSHandler

Handles SNMP classification and identification for UPSs.
Use this API for SNMP-related discovery.

UPSHandler - classifyAndIdentify()

Classifies and identifies the UPSs.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

UserCriteria

UserCriteria API enables you to create, modify, or delete user criteria records using scripts. To use this class in a scoped application, use the sn_uc namespace identifier. The User Criteria Scoped API plugin (ID: com.glideapp.user_criteria.scoped.api) should be enabled to access the UserCriteria API.

Scoped UserCriteria - create(Object columnValues, Boolean standardUpdate)

Creates a user criteria with specified values in the user_criteria table. Values specified in columnValues override the values provided via setters.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnValues</td>
<td>Object</td>
<td>Key and value pairs for a column and its value.</td>
</tr>
<tr>
<td>standardUpdate</td>
<td>Boolean</td>
<td>Set to true to enable the running of engines and workflow.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>sys_id of the created user criteria.</td>
</tr>
</tbody>
</table>

### Example

```javascript
var uc = new sn_uc.UserCriteria();
uc.setCompanies("31bea3d53790200044e0bfc8bcbe5dec,0c441abbc6112275000025157c651c89");
uc.setActive(true);
uc.setUsers("31bea3d53790200044e0bfc8bcbe5dec,0c441abbc6112275000025157c651c89");
var UserCriteriaId = UserCriteria.create();
gs.info(UserCriteriaId);
```

**Output**

```
41bea3d53790200044e0bfc8bcbe5dec
```

### Scoped UserCriteria - deleteRecord()

Deletes the current user criteria.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If true, the user criteria is deleted.</td>
</tr>
<tr>
<td></td>
<td>If false, no user criteria is found to delete.</td>
</tr>
</tbody>
</table>

#### Example

```javascript
var uc = new sn_uc.UserCriteria("31bea3d53790200044e0bfc8bcbe5dec");
uc.deleteRecord();
```
Output

true

**Scoped UserCriteria - read(String[] columns)**

Displays the mapping for the attribute and value pairs of the catalog item.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columns</td>
<td>String</td>
<td>Array of catalog item attributes.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Mapping for the attribute and value pairs of the catalog item.</td>
</tr>
</tbody>
</table>

**Example**

```java
var uc = new sn_uc.UserCriteria();
uc.read(\"name", \"applies_to\"); 
```

Output

name: My Catalog Item; 
applies_to: catalog_item;

**Scoped UserCriteria - setActive(Boolean active)**

Specifies if the user criteria is active.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>Boolean</td>
<td>If true, the user criteria is active. If false, the user criteria is inactive.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
Example

```javascript
var uc = new sn_uc.UserCriteria();
uc.setActive(true);
```

**Scoped UserCriteria - setAdvanced(boolean advanced)**

Specifies if the user criteria has an advanced script.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>advanced</td>
<td>Boolean</td>
<td>If true, the user criteria has an advanced script.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If false, the user criteria does not have an advanced script.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var uc = new sn_uc.UserCriteria();
uc.setAdvanced(true);
```

**Scoped UserCriteria - setCompanies(String companies)**

Sets the company property for the user criteria.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>companies</td>
<td>String</td>
<td>Comma-separated list of the company sys_ids to be set for the user criteria.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
Scoped UserCriteria - setDepartments(String departments)

Sets the department property for the user criteria.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>departments</td>
<td>String</td>
<td>Comma-separated list of the department sys_ids to be set for the user criteria.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var uc = new sn_uc.UserCriteria();
uc.setDepartments("31bea3d53790200044e0bfc8bcbe5dec,0c441abbc6112275000025157c651c89");
```

Scoped UserCriteria - setGroups(String groups)

Sets the group property for the user criteria.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>String</td>
<td>Comma-separated list of the group sys_ids to be set for the user criteria.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var uc = new sn_uc.UserCriteria();
uc.setGroups("31bea3d53790200044e0bfc8bcbe5dec,0c441abbc6112275000025157c651c89");
```
**Scoped UserCriteria - setLocations(String locations)**

Sets the location property for the user criteria.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>locations</td>
<td>String</td>
<td>Comma-separated list of the location sys_ids to be set for the user criteria.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```java
var uc = new sn_uc.UserCriteria();
uc.setLocations("31bea3d53790200044e0bfc8bcbe5dec,0c441abbc6112275000025157c651c89");
```

**Scoped UserCriteria - setMatchAll(Boolean match_all)**

Sets the match_all property for the user criteria.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| None     | match_all       | If true, all conditions set as properties for the user criteria should be fulfilled.  
            |                  | If false, at least one condition set as a property for the user criteria should be fulfilled. |

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```java
var uc = new sn_uc.UserCriteria();
uc.setMatchAll(true);
```
**Scoped UserCriteria - setName(String name)**

Sets the name property for the user criteria.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the user criteria.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```javascript
var uc = new sn_uc.UserCriteria();
uc.setName("Property1");
```

**Scoped UserCriteria - setRoles(String roles)**

Sets the role property for the user criteria.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roles</td>
<td>String</td>
<td>Comma-separated list of the role sys_ids to be set for the user criteria.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```javascript
var uc = new sn_uc.UserCriteria();
uc.setRoles("31bea3d53790200044e0bfc8bcbe5dec,0c441abbc6112275000025157c651c89");
```

**Scoped UserCriteria - setScript(String script)**

Sets the script for the user criteria.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>script</td>
<td>String</td>
<td>Script to be set for the advanced user criteria.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var uc = new sn_uc.UserCriteria();
uc.setScript("function scriptTest() {
  var retVal;
  if (gs.getUser().getRecord().getDisplayValue('department') == 'Product Management') {
    retVal = true;
  } else {
    retVal = false;
  }
  return retVal;
}");
```

Scoped UserCriteria - setUsers(String users)
Sets the user property for the user criteria.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>users</td>
<td>String</td>
<td>Comma-separated list of the user sys_ids to be set for the user criteria.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var uc = new sn_uc.UserCriteria();
uc.setUsers("31bea3d5379020044e0bf8bcbe5dec,0c441abbc611227500025157c651c89");
```
**Scoped UserCriteria - update(Object columnValues, String reason)**

Updates the current catalog item with the specified values.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnValues</td>
<td>Object</td>
<td>Mapping for the column name and the value pairs.</td>
</tr>
<tr>
<td>reason</td>
<td>String</td>
<td>Reason for updating the catalog item.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Returns the sys_id of the created user criteria.</td>
</tr>
</tbody>
</table>

### Example

```javascript
var uc = new sn_uc.UserCriteria();
uc.update("name": "Updated name", "The existing name is not relevant. Setting a relevant name");
```

**Output**

```
31bea3d53790200044e0bfc8bcbe5dec
```

**Scoped UserCriteria - UserCriteria()**

Creates an instance of the UserCriteria class.

```javascript
var uc = new sn_uc.UserCriteria();
```

**Scoped UserCriteria - UserCriteria(String sys_id)**

Creates an instance of the UserCriteria class with the specified sys_id.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>String</td>
<td>sys_id of the user criteria.</td>
</tr>
</tbody>
</table>

```javascript
var uc = new sn_uc.UserCriteria("31bea3d53790200044e0bfc8bcbe5dec");
```
VariablePoolQuestionSetJS - Scoped

VariablePoolQuestionSetJS API enables you to use Variable Pool Question Set. To use this class in a scoped application, use the sn_sc namespace identifier. The Service Catalog Scoped API plugin (com.glideapp.servicecatalogscoped.api) that is enabled by default is required to access the VariablePoolQuestionSetJS API.

VariablePoolQuestionSetJS - VariablePoolQuestionSetJS()

Creates an instance of the VariablePoolQuestionSet class.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example:

```javascript
var variablePool=new sn_sc.VariablePoolQuestionSetJS();
```

VariablePoolQuestionSetJS - getFlatQuestions()

Returns the array of questions associated with the cart item ids specified.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Object pointing to the current cart details.</td>
</tr>
</tbody>
</table>

Example:

```javascript
var cart=new sn_sc.VariablePoolQuestionSetJS();
cart.setCartID("9bf16af87110300318d05a88cb0b49");
cart.load();
console.log(cart.getFlatQuestions());
```

Output:

```javascript
   displayValue
   : ""
sys_id
   : "90b72d4b4f7b4200086eed18110c701"
type
```
VariablePoolQuestionSetJS - load()

Loads the question set.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
VariablePoolQuestionSetJS - setCartID(String id)

Sets the cart item ids of the variable pool.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example:

```javascript
var cart=new sn_sc.VariablePoolQuestionSetJS();
cart.load();
cart.setCartID("9bf16af87110300318d05a888cb0b49");
```

VMUtils

Provides utility functions for discovery and provisioning of virtual machines.

The VMUtils class is available in server-side scripts to convert between UUID and Correlation ID formats.

VMUtils - turnCorrelationIdToUuid(String correlationId)

Converts a correlation ID into a UUID for a virtual machine in the ServiceNow system.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlationId</td>
<td>String</td>
<td>The correlation ID to convert</td>
</tr>
</tbody>
</table>

Example:

```javascript
var cart=new sn_sc.VariablePoolQuestionSetJS();
cart.load();
cart.setCartID(9bf16af87110300318d05a888cb0b49);`
returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The UUID, or an empty string if invalid.</td>
</tr>
</tbody>
</table>

```
var vmu = new VMUtils();
gs.print(vmu.turnCorrelationIdToUuid('42 10 82 82 62 35 ca 68-b5 1c 1e f8 5c 0a 0d 5b'));
```

**VMUtils - turnUuidToCorrelationId(String uuid)**

Converts a VMware universal unique identifier (UUID) to a format that matches the original format in the ServiceNow system.

vCenter works with UUIDs where the ESXi Linux console worked with this format.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>String</td>
<td>The UUID to convert.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The identifier converted to the ServiceNow format.</td>
</tr>
</tbody>
</table>

```
var vmu = new VMUtils();
vmu.turnUuidToCorrelationId('13eb78d0-d504-11e1-9b23-0800200c9a66');
```

**WalkWorkflow**

Walks the workflow and gets the list of successors for each activity along with a shortest path order that indicates the activity depth in the workflow.

**WalkWorkflow - computeFullSequences()**

Computes the full sequence for every path.

Does not combine paths at Join activities. After calling this method, call `getSequences()` to yield the computed sequences.
### Walkworkflow - `computeSequences()`

Computes the sequences of activities.

All sequences leading up to a join end at the join and then a single sequence from the join is computed. This ensures that all sequences can be walked up to a join, and then the sequence can be walked from the join. After calling this method a call to `getSequences()` will yield the computed sequences.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### Walkworkflow - `dump()`

Outputs the internal state of this class to `gs.print()`.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### Walkworkflow - `dumpSequences()`

Outputs the sequences internal to this class to `gs.print()`.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Walkworkflow - dumpShortestPathOrders( description)**

Outputs the shortest path orders internal to this object to `gs.print()`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Walkworkflow - getActivitiesAtOrder(Number order)**

Returns all the activities at the specified shortest path order.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>Number</td>
<td>The order number the client is interested in.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Array</td>
<td>An array of activity sys_ids at the specified order. If an empty array is returned, there are no more activities.</td>
</tr>
</tbody>
</table>

**Walkworkflow - getSequences()**

Gets the activity sequences.

The `walk()` method must have been called prior to calling this method.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A string of 1-delimited sequences. Each sequence is a comma-delimited list of activity ids (table wf_workflow_activity).</td>
</tr>
</tbody>
</table>

### Walkworkflow - walk(Boolean fullSequences)

Walks the workflow and sets up the activity shortest path ordering. After calling this method the client would call the various accessor methods to obtain relevant data about the workflow.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullSequences</td>
<td>Boolean</td>
<td>(Optional) If true, computes the full sequences. Default is false.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### Walkworkflow - Walkworkflow(GlideRecord workflowVersion)

Constructor for Walkworkflow class.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowVersion</td>
<td>GlideRecord</td>
<td>GlideRecord on table wf_workflow_version of the workflow version to analyze.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow</td>
<td>Workflow object just created.</td>
</tr>
</tbody>
</table>

### WFActivityHandler

The base class for all workflow activities.
As the base class for all workflow activities, this code always executes as part of an activity. When developing your own workflow activities, create your script object by deriving from this class and overriding methods as necessary to get the functionality you want. Client code must override method `onExecute()` to perform meaningful activity processing.

**WFActivityHandler - debug(String msg, String args)**
Logs a debug message.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to send to the log.</td>
</tr>
<tr>
<td>args</td>
<td>String</td>
<td>String to send to the log.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**WFActivityHandler - debug(String msg, Array args)**
Logs a debug message.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to send to the log.</td>
</tr>
<tr>
<td>args</td>
<td>Array</td>
<td>Array of values to send to the log</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**WFActivityHandler - generate(String activityId, String order, GlideDateTime startDate, Boolean noCreateFlag)**
Handles pre-generation of tasks and approvals (and other objects as needed).

Allows objects to be generated at the start of a workflow in a `pending` state and not requested until the workflow gets to the activity, thereby providing a preview of the work to be done in the future. To pre-generate objects, an activity’s `_generate()` method is called. This method typically returns a pre-generation object that is saved in the workflow scratchpad. (For example, the object might be an array of approval ids that were generated.) When the activity runs `onExecute` it may request the pre-generation object and determine if the approvals have already been created in a pending state or if it needs to create the approvals in the `onExecute` method. Internal method `_generate` needs to be overridden and needs to return an object with
information that it can retrieve later on when the activity’s onExecute() method is called. The object must contain a duration property that contains the duration of the task(s) or approval(s) that were generated by the activity. See activity Generate for example usage.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activityId</td>
<td>String</td>
<td>sys_id of this running activity (table wf_activity).</td>
</tr>
<tr>
<td>order</td>
<td>String</td>
<td>The order number associated with the task or approval.</td>
</tr>
<tr>
<td>startAtDspValue</td>
<td>GlideDateTime</td>
<td>The starting time for the task or approval</td>
</tr>
<tr>
<td>noCreateFlag</td>
<td>Boolean</td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The duration property of the _generate return value is returned by this method.</td>
</tr>
</tbody>
</table>

WFActivityHandler - info(String msg, String args)

Logs an information message.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to send to the log.</td>
</tr>
<tr>
<td>args</td>
<td>String</td>
<td>String to send to the log.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

WFActivityHandler - info(String msg, Array args)

Logs an information message.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to send to the log.</td>
</tr>
<tr>
<td>args</td>
<td>Array</td>
<td>Array of values to send to the log</td>
</tr>
</tbody>
</table>
### WFActivityHandler - js(String str)

Evaluate activity variables in the script. Used to evaluate strings that contain el such as `${some script}` or strings starting with `javascript:`. See almost any existing activity for example usage.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>String</td>
<td>String that possibly contains embedded el.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The evaluated string.</td>
</tr>
</tbody>
</table>

### WFActivityHandler - onCancel()

Event handler for `cancel` event. The base class for the activity script sets this activity to be cancelled. Derived classes (activities) can override this method if additional processing is required to cancel this activity. Activity **Manual Approvals** is an example of overriding this method to perform additional processing to cancel this activity.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

### WFActivityHandler - onExecute()

Virtual method. Activity subclasses must override this method to perform work appropriate to the activity.
WFActivityHandler - runScript(String script)

Enables activities to run a script contained in an activity variable of type script.
Example activities that use this method include If, Wait for condition, and Approval - User.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>script</td>
<td>String</td>
<td>String containing valid Javascript.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable</td>
<td>If the script set the workflow variable answer then this value is returned. If not, then the result of the script execution is returned.</td>
</tr>
</tbody>
</table>

WFActivityHandler - setActivityOutput()

Sets the activity output property.
See activity SOAP Message.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

WFActivityHandler - setResultFailed(String reason)

Sets the activity result as failed with an optional reason string.
See Activity SOAP Message.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reason</td>
<td>String</td>
<td>(Optional) Description of the reason this activity failed.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**WFAActivityHandler - setResultSucceeded()**

Sets the result of this activity as successful.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**WFAActivityHandler - warn(String msg, String args)**

Logs a warning message.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to send to the log.</td>
</tr>
<tr>
<td>args</td>
<td>String</td>
<td>String to send to the log</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**WFAActivityHandler - warn(String msg, Array args)**

Logs a warning message.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to send to the log.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>----------------------</td>
</tr>
<tr>
<td>args</td>
<td>Array</td>
<td>Array of values to send to the log</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**WindowsOSNameHelper**

Handles the formatting of Windows OS names.

Use in any server-side discovery scripts where you need to format OS names.

**WindowsOSNameHelper - formatWindowsOSName(String name)**

Formats the given Windows OS name.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name to format</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The formatted name</td>
</tr>
</tbody>
</table>

**WindowsOSNameHelper - makeOSChoiceValid(String value)**

Ensures that the OS choice is valid.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>String</td>
<td>The value to validate</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**WindowsOSNameHelper - osCleanupName(String name)**

Cleans up the specified Windows OS name by stripping out incorrect characters, incorrect capitalization, extra spaces, or trailing spaces.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The name to clean up.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The cleaned up name</td>
</tr>
</tbody>
</table>

Workflow

Workflow scripts provide an interface with the workflow engine.
Use these methods to manipulate workflows.

Workflow - broadcastEvent(String contextId, String eventName)

Sends the specified event (message) into the workflow context to pass along to the executing activities.

Typical use of this method is to enable activities that wait for some action to occur before proceeding. For additional information on using broadcastEvent, refer to Workflow event-specific functions.

For a list of the available base instance events, refer to Workflow events in the base system.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextId</td>
<td>String</td>
<td>Context ID.</td>
</tr>
<tr>
<td>eventName</td>
<td>String</td>
<td>Name of the event.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
//where current is a task record with a workflow context
var wf = new Workflow().getRunningFlows(current);
while(wf.next()) {
    new Workflow().broadcastEvent(wf.sys_id, 'resume');
}
```

Workflow - cancel(GlideRecord record)

Cancels all running workflows on this record by broadcasting the cancel event to activities in all running workflows on this record.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>GlideRecord</td>
<td>GlideRecord on any table. All workflows running on this record will be cancelled.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
//get workflow helper
var workflow = new Workflow();
//cancel all the workflows, where current is a task record with a workflow context
workflow.cancel(current);
gs.addInfoMessage(gs.getMessage("Workflows for {0} have been cancelled", current.getDisplayValue()));
```

### Workflow - cancelContext(GlideRecord context)

Cancels this running context by broadcasting a **cancel** event to activities in this workflow context.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>GlideRecord</td>
<td>GlideRecord of the running context to cancel.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
// If a workflow has started for this item, cancel it, where current is a task record with a workflow context
if ((current.stage == 'Request Cancelled') && current.context && !current.context.nil()) {
    var w = new Workflow();
    var gr = new GlideRecord('wf_context');
    if (gr.get(current.context))
        w.cancelContext(gr);
```
Workflow - deleteWorkflow(GlideRecord current)

Deletes all the workflows on the record.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current</td>
<td>GlideRecord</td>
<td>GlideRecord for which the caller wants to delete all workflows. This can be any record on any table.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
//where current is a task record with a workflow context
var wkfw = new Workflow();
wkwf.deleteWorkflow(current);
```

Workflow - fireEvent(GlideRecord eventRecord, String eventName)

Fires the named event on the input record.

Used in Activities Approval Coordinator, Timer, Lock, and some others.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventRecord</td>
<td>GlideRecord</td>
<td>The event record.</td>
</tr>
<tr>
<td>eventName</td>
<td>String</td>
<td>The name of the event to send to the executing workflow.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
///where current is a task record with a workflow context
var w = new Workflow();
```
Workflow - `fireEventById(String eventRecordId, String eventName)`

Fires the named event on the record specified by record ID.

Used in Activities: Approval Coordinator, Timer, Lock, and some others.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventRecordId</td>
<td>String</td>
<td>The sys_id of the glide record.</td>
</tr>
<tr>
<td>eventName</td>
<td>String</td>
<td>The name of the event to send to the executing workflow.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var wkfw = new Workflow();
wkw fireEventById('f2400ec10b0a3c1c00ca5bb5c6fae427','Timer');
```

Workflow - `getContexts(GlideRecord record)`

Gets all workflow contexts for the input record.

The input record is any record on any table for which the caller wants the running workflow contexts.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>GlideRecord</td>
<td>GlideRecord for which the caller wants a list of all workflow contexts.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>GlideRecord on table wf_context filtered for all workflow contexts for the input record (in any state, e.g. running, cancelled, finished).</td>
</tr>
</tbody>
</table>

```javascript
//where current is a task record with a workflow context
```
var wkfw = new Workflow();
gs.print(wkfw.getContexts(current).started);

Workflow - getEstimatedDeliveryTime(String workflowId)

Gets the estimated time for a workflow to complete.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowId</td>
<td>String</td>
<td>Sys_id of the workflow (table wf_workflow) to get the estimated run time.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Display value from a GlideDuration (e.g., 3 days), or blank if unknown.</td>
</tr>
</tbody>
</table>

var wkfw = new Workflow();
gs.print(wkfw.getEstimatedDeliveryTime('b99a866a4a3623120074c033e005418f'));
2 Days

Workflow - getEstimatedDeliveryTimeFromWFVersion(GlideRecord wfVersion)

Get the estimated elapsed execution time for the workflow version.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wfVersion</td>
<td>GlideRecord</td>
<td>GlideRecord on table wf_workflow_version of a specific workflow version for which the caller wants the estimated duration of executing.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Display value from a GlideDuration (e.g., 3 days), or blank if unknown.</td>
</tr>
</tbody>
</table>
// where current is a task record with a workflow context
var wkfw = new Workflow();
var context = wkfw.getContexts(current);

gs.print(wkfw.getEstimatedDeliveryTimeFromWFVersion(context.wf_version));

Workflow - getReturnValue(String workflowID, Number amount, Boolean result)

Gets the appropriate workflow return value for the input workflow ID. This is either the workflow
checked out by the current user or the published workflow with the most recent date.
This is either the workflow checked out by the current user or the published workflow with the most
recent date. This method is available starting with the Fuji release.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowID</td>
<td>String</td>
<td>The sys_id of the workflow (table wf_workflow)</td>
</tr>
<tr>
<td>amount</td>
<td>Number</td>
<td>amount</td>
</tr>
<tr>
<td>result</td>
<td>Boolean</td>
<td>True, if true</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| ???  | The return value of the workflow as specified by
       | the Return Value activity. Workflows without a
       | Return Value activity return a null value.      |

var wkfw = new Workflow();
wkfw.getReturnValue('context');

Output:

*** Script: b99a866a4a3623120074c033e005418f

Workflow - getRunningFlows(GlideRecord record)

Gets all the currently running workflow contexts for the input record.
The input record is any record on any table for which the caller wants the running workflow contexts.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>GlideRecord</td>
<td>GlideRecord of the record for which the caller wants a list of all running workflows.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>GlideRecord on table wf_context and filtered for all executing workflow contexts.</td>
</tr>
</tbody>
</table>

```javascript
//where current is a task record with a workflow context
var wf = new Workflow().getRunningFlows(current);
while(wf.next()) {
    new Workflow().broadcastEvent(wf.sys_id, 'pause');
}
```

Workflow - `getVersion(String workflowID)`

Gets the appropriate workflow version for the input workflow ID. This is either the workflow checked out by the current user or the published workflow with the most recent date.

This is either the workflow checked out by the current user or the published workflow with the most recent date.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowID</td>
<td>String</td>
<td>The sys_id of the workflow (table wf_workflow)</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var wkfw = new Workflow();
wkfw.getVersion('b99a866a4a3623120074c033e005418f');
```

Workflow - `getVersionFromName(String workflowName)`

Returns the appropriate workflow version for the input workflow name.

See `getVersion()` for more information.
### Workflow - getWorkflowFromName(String workflowName)

Returns the sys_id of the workflow associated with the specified workflow name.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowName</td>
<td>String</td>
<td>Name of the workflow.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The sys_id of the workflow associated with the passed in name.</td>
</tr>
</tbody>
</table>

```javascript
var wkfw = new Workflow();
wkwf.getVersionFromName('Emergency Change');
```

### Workflow - hasWorkflow(GlideRecord record)

Determines if a specified record has any workflow contexts associated to it. This includes running and completed workflow contexts.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>GlideRecord</td>
<td>GlideRecord under scrutiny. This GlideRecord can be from any table.</td>
</tr>
</tbody>
</table>

```javascript
var wflw = new Workflow();
gs.print(wflw.getWorkflowFromName('Emergency Change'));
```
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True, if record has associated workflow; otherwise, returns False.</td>
</tr>
</tbody>
</table>

```javascript
var wkfw = new Workflow();
gs.print(wkfw.hasWorkflow('f2400ec10b0a3c1c00ca5bb5c6fae427'));
false
```

**Workflow - restartWorkflow(GlideRecord current, Boolean maintainStateFlag)**

Recalculates the approvals and tasks for a workflow by adding new approvals and tasks while not resetting current approvals and tasks.

Use this script to add an additional company to a change request, without resetting current approvals for companies already in the workflow.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current</td>
<td>GlideRecord</td>
<td>GlideRecord of the record this workflow is executing. This can by any record on any table.</td>
</tr>
<tr>
<td>maintainStateFlag</td>
<td>Boolean</td>
<td>If True, maintains all approvals and tasks in their current state.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
//mark all existing approvals for the change as 'cancelled' and
//where current is a task record with a workflow context
new WorkflowApprovalUtils().cancelAll(current, comment);
new Workflow().restartWorkflow(current);
```

**Workflow - runFlows(GlideRecord record, String operation)**

Runs all workflows for a given record in a given table and its descendant tables. Sample usage can be seen in the Script Includes "SNC - Run parent workflows", and "SNC - Run parent workflows (Approval)". 
## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>GlideRecord</td>
<td>The record to run workflows against.</td>
</tr>
<tr>
<td>operation</td>
<td>String</td>
<td>Database operation. One of insert, update, delete.</td>
</tr>
</tbody>
</table>

## Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var wkfw = new Workflow();
wkwf.runFlows('f2400ec10b0a3c1c00ca5bb5c6fae427','update');
```

### Workflow - startFlow(String workflowId, GlideRecord current, String operation, Array vars)

Starts a specified workflow. See script include WorkflowScheduler and Business Rule "Start Workflow" on table sc_req_item for examples of use.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowId</td>
<td>String</td>
<td>The sys_id of the workflow to start. This sys_id refers to table wf_workflow.</td>
</tr>
<tr>
<td>current</td>
<td>GlideRecord</td>
<td>The record to use as current in this workflow. This is normally from the Table field of the workflow properties for this workflow.</td>
</tr>
<tr>
<td>operation</td>
<td>String</td>
<td>The operation to perform on <strong>current</strong>. Possible values: insert, update, delete.</td>
</tr>
<tr>
<td>vars</td>
<td>Array</td>
<td>Collection of variables to add to the workflow.</td>
</tr>
</tbody>
</table>

```javascript
var w = new Workflow();
var context = w.startFlow(id, current, current.operation(), getVars());
```
Workflow - startFlowFromContextInsert(GlideRecord context, String operation)

Helper method for business rule Auto start on context.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>GlideRecord</td>
<td>GlideRecord on table wf_context of a new record (the &quot;current&quot; record in the business rule).</td>
</tr>
<tr>
<td>operation</td>
<td>String</td>
<td>Database operation being performed. One of insert, update, delete.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
//where current is a task record with a workflow context
current.name = current.workflow_version.name;
current.started_by.setValue(gs.userID());

if (gs.nil(current.id)) {
    var gr = new GlideRecord('wf_workflow_execution');
gr.name = current.name;
gr.insert();

current.table = 'wf_workflow_execution';
current.id = gr.sys_id;
}

var wf = new Workflow();
wf.startFlowFromContextInsert(current, current.operation())
```

Workflow - startFlowRetroactive(String workflowId, Number retroactiveMSEcs, GlideRecord current, String operation, Array, ???)

Used by business rule Start Workflow on table task_sla. This starts a workflow and the extra arguments to this method are used by activity "Timer" to pause the execution of the workflow for some duration.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowID</td>
<td>String</td>
<td>The sys_id of the workflow to start. This sys_id refers to table wf_workflow.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>retroactiveMSecs</td>
<td>Number</td>
<td>Delay in milliseconds used by Activity Timer.</td>
</tr>
<tr>
<td>current</td>
<td>GlideRecord</td>
<td>GlideRecord of the record to use as current in this workflow. This is normally from the Table field of the workflow properties for this workflow</td>
</tr>
<tr>
<td>operation</td>
<td>String</td>
<td>Database operation being performed. One of insert, update, delete.</td>
</tr>
<tr>
<td>vars</td>
<td>Array</td>
<td>Collection of variables to add to the workflow.</td>
</tr>
<tr>
<td>withSchedule</td>
<td>???</td>
<td>Schedule used by Activity Timer.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>A GlideRecord on table wf_context on the inserted record for this newly created workflow context.</td>
</tr>
</tbody>
</table>

```javascript
// is this a retroactive start?
```
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

var w = new Workflow();

**Workflow**

The scoped Workflow API provides methods that can be used in an activity definition script. There are no constructors for creating an instance of a scoped workflow object. Instead, use the global `workflow` object available in activity scripts. This workflow object is available in any script location inside a workflow.

**Scoped Workflow - debug(String message, Object args)**

Adds a debug message to the log.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to add to the log.</td>
</tr>
<tr>
<td>args</td>
<td>Object</td>
<td>Arguments to add to the message.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message added to the log.</td>
</tr>
</tbody>
</table>

var loggedMessage = workflow.debug("All is well");

**Scoped Workflow - error(String message, Object args)**

Adds an error message to the log.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to add to the log.</td>
</tr>
<tr>
<td>args</td>
<td>Object</td>
<td>Arguments to add to the message.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The logged message</td>
</tr>
</tbody>
</table>

```javascript
var loggedMessage = workflow.error("An error has occurred.");
```

Scoped Workflow - `getVariable(String name)`

Returns the specified variable's value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The variable name</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The variable's value</td>
</tr>
</tbody>
</table>

```javascript
var value = workflow.getVariable("task");
```

Scoped Workflow - `info(String message, Object args)`

Adds an informational message to the log.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to add to the log.</td>
</tr>
<tr>
<td>args</td>
<td>Object</td>
<td>Arguments to add to the message.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.

ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The message that is logged.</td>
</tr>
</tbody>
</table>

```javascript
var loggedMessage = workflow.info("All is well");
```

Scoped Workflow - inputs

Returns the workflow variables.

Field

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputs</td>
<td>Object</td>
<td>Workflow variables as name value pairs.</td>
</tr>
</tbody>
</table>

```javascript
var user = workflow.inputs.u_user;
```

Scoped Workflow - name()

Returns the workflow name.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The workflow name</td>
</tr>
</tbody>
</table>

```javascript
var name = workflow.name();
```

Scoped Workflow - removeVariable(String name)

Removes the specified variable from the workflow.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The variable name</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var value = workflow.removeVariable("task");
```

Scoped Workflow - result

Returns the workflow's result.

Field

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>String</td>
<td>Workflow's result</td>
</tr>
</tbody>
</table>

```javascript
var wfResult = workflow.result;
```

Scoped Workflow - scratchpad()

Returns the workflow's scratchpad object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The scratchpad object.</td>
</tr>
</tbody>
</table>

```javascript
var scratchpad = workflow.scratchpad();
```
**Scoped Workflow - setResult(String result)**

Sets the workflow's result.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>String</td>
<td>The workflow's result</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
workflow.setResult("Success");
```

**Scoped Workflow - setVariable(String name, Object value)**

Sets the specified variable to the specified value.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The variable name</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>The value to be assigned to the variable.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```
workflow.setVariable("task", "terrible");
```

**Scoped Workflow - warn(String message, Object args)**

Adds a warning message to the log.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>The message to add to the log.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>args</td>
<td>Object</td>
<td>Arguments to add to the message.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The logged message</td>
</tr>
</tbody>
</table>

```javascript
var loggedMessage = workflow.warn("Check your permissions.");
```

**WorkflowDuration**

This class calculates the duration (in seconds) based on the variables of a workflow activity. It is an interface between `Workflow Timer()` and `DurationCalculator()`.

**WorkflowDuration - addSeconds(GlideRecord record, Number amount)**

Adds the number of seconds to the specified start date/time.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>GlideRecord</td>
<td>description</td>
</tr>
<tr>
<td>amount</td>
<td>Number</td>
<td>amount</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True, if object was successfully created.</td>
</tr>
</tbody>
</table>

**WorkflowDuration - calculate(GlideRecord record)**

Calculates the number of seconds and the due date.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>GlideRecord</td>
<td>The record that contains the fields with the schedule, timezone, and date/time information. This is almost always <code>activity.vars.__var_record__</code>.</td>
</tr>
</tbody>
</table>
WorkflowDuration - calculateTimeLeft(GlideRecord record)
Calculates the number of seconds remaining to the specified end date/time.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>GlideRecord</td>
<td>The record that contains the fields with the schedule, timezone, and date/time information. This is almost always activity.vars.<strong>var_record</strong>.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

WorkflowDuration - getEndDateTime( )
Returns the end date/time set by a call to calculate(record).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideDateTime</td>
<td>The end date/time.</td>
</tr>
</tbody>
</table>

WorkflowDuration - getSeconds( )
Returns the seconds value that was set by a call to calculate(record).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>The number of seconds set by a call to calculate(record).</td>
</tr>
</tbody>
</table>

**WorkflowDuration - getTotalSeconds()**

Returns the totalSeconds value that was set by a call to calculate(record).

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**WorkflowDuration - setEndDateTime(String dt)**

Sets the end date/time to use when calculating the remaining time.

To convert the value into the GlideDateTime internal format, use GlideDateTime.getValue().

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dt</td>
<td>String</td>
<td>The end date/time to use.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**WorkflowDuration - setEndDateTime(GlideDateTime dt)**

Sets the end date/time to use when calculating the remaining time.

To convert the value into the GlideDateTime internal format, use GlideDateTime.getValue().

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dt</td>
<td>GlideDateTime</td>
<td>The end date/time to use.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**WorkflowDuration - setStartDateTime(String dt)**

Sets the start date/time to use in the calculations.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dt</td>
<td>String</td>
<td>The start date/time to use, in GMT.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**WorkflowDuration - setStartDateTime(GlideDateTime dt)**

Sets the start date/time to use in the calculations.

To convert the value into the GlideDateTime internal format, use GlideDateTime.getValue().

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dt</td>
<td>GlideDateTime</td>
<td>The start date/time to use.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**WorkflowDuration - setUsedSecs(Number secs)**

Sets the used seconds compensation, where the number of seconds is the number of seconds inside of any schedule.

Can be a -ve number, which extends the duration.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>secs</td>
<td>Number</td>
<td>The number of used seconds to set.</td>
</tr>
</tbody>
</table>
**WorkflowDuration** - setWorkflow(Object schedule, Object timezone)

Sets the workflow schedule/timezone. Used for `schedule_type workflow_schedule` and `timezone_type workflow_timezone`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>schedule</td>
<td>Object</td>
<td>The schedule to use. Usually from the workflow context.schedule.</td>
</tr>
<tr>
<td>timezone</td>
<td>Object</td>
<td>The timezone to use. Usually from the workflow context.timezone.</td>
</tr>
</tbody>
</table>

**WorkObjectModelManager**

The `WorkObjectModelManager` class provides a way to query the workflow model, to step backwards and forwards between specified `wf_history` items, and to query the history for activity and transition-specific information.

The interaction with the `WorkObjectModelManager` class is through the `ActivityHistoryRecord` data object. The `ActivityHistoryRecord` is an inner class and is only used while interacting with `WorkObjectModelManager`.

To acquire the executed history of the workflow activities:

```javascript
var model = new WorkObjectModelManager('myContextId');
var activities = model.getExecutedHistory();
```

To output the playback of the workflow:

```javascript
var model = new WorkObjectModelManager('myContextId');
model.getExecutedHistory();
model.playBack();
```

At this time, the playBack is required to load the executed transitions. This will also play a role in walking backwards on a model to rollback to a specific activity.

To see the details of the cached model:

```javascript
var model = new WorkObjectModelManager('myContextId');
model.getExecutedHistory();
```
WorkflowModelManager - getActivityHistoryRecordById(String haRecordSysId)

Retrieves the history activity that is cached by the `wf_history.sys_id` provided in the argument.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>haRecordSysId</td>
<td>String</td>
<td>The sys_id of the activity history (table <code>wf_history</code>).</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivityHistoryRecord</td>
<td>The ActivityHistoryRecord JavaScript object requested.</td>
</tr>
</tbody>
</table>

WorkflowModelManager - getAllTransitionedIntoActivity(Object description)

Retrieves the history activities that executed and transitioned into the one represented by the `sys_id` in the argument.

The next status is based on the `wf_activity.sys_id` associated with the activity represented in the haRecord existing as a TO in a transition associated with any ActivityHistoryRecords that executed in the workflow's history. (This differs from `getPreviousByTransition`, which only returns TO transitions that come before the haRecord in the execution sequence (by time).) The return value is a collection of ActivityHistoryRecords that identify the argument haRecord.wfaId as their TO activity. The return values are based on all transitions in the executed history collection that transition To get the activity that executed prior to this activity in time use `getPreviousByExecutedOrder`.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>Object</td>
<td>ActivityHistoryRecord JavaScript object.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Array</td>
<td>Array of ActivityHistoryRecord JavaScript objects.</td>
</tr>
</tbody>
</table>
The next status is based on the `wf_activity.sys_id` associated with the activity represented in the haRecord existing as a TO in a transition associated with any ActivityHistoryRecords that executed in the workflow's history. (This differs from `getPreviousByTransition`, which returns only TO transitions that come before the haRecord in the execution sequence (by time).) The return value is a collection of ActivityHistoryRecords that identify the argument `haRecord.wfaId` as their TO activity.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>haRecordSysId</td>
<td>String</td>
<td>The sys_id of the activity history (table <code>wf_history</code>).</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Array</td>
<td>Array of <code>ActivityHistoryRecord</code> JavaScript objects requested.</td>
</tr>
</tbody>
</table>

```javascript
var model = new WorkflowModelManager('a143585c3b001000dada82c09ccf3d44');
model.getExecutedHistory();
var activity = model.begin;
gs.print('activity: ' + activity.wfaName + ', transitions: ' + activity.transitions.length);
while( activity != null){
gs.print('activity: ' + activity.wfaName + ', transitions: ' + activity.transitions.length);
  var parents =
model.getAllTransitionedIntoActivity(activity);
  for( var i = 0; i < parents.length; i++ ){
gs.print(' ---------------  parent activity: ' + parents[i].wfaName );
  }
  activity = model.getNextByExecutedOrder( activity );
}
```

**WorkflowModelManager - getExecutedHistory()**

This is the worker method that must be called to initialize the `WorkflowModelManager` object. In the process of initializing the `WorkflowModelManager` object, `getExecutedHistory()` creates and populates the `activityHistoryRecord` data object array (which is a member of the `WorkflowModelManager` class). Most `WorkflowModelManager` methods return a subset of the `activityHistoryRecord` array. Your script can then call one or more of the `activityHistoryRecord` methods to accomplish further work.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Array</td>
<td>An ordered array of sys_ids from table wf_history. It is unlikely client code would use the return value; it is kept internally to this object for use by subsequent method calls.</td>
</tr>
</tbody>
</table>

```javascript
var model = new WorkflowModelManager('myContextId');
var activities = model.getExecutedHistory();
var current = model.getActivityHistoryRecordById('d6681d573b130000dada82c09ccf3d10');
```

### WorkflowModelManager - `getFinalExecutedActivityList()`

Queries the `wf_history` table by context and retrieves all the activities executed in the workflow given by the context set in the construction of this object.

This function produces a list of executed activities in the exact order each activity passed through the server side `ActivityManager.java` using the new `activity_index` to force the order coming out of the database. On its this call will not give the full picture; it needs to load and map the transitions.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Array</td>
<td>Array of <code>ActivityHistoryRecord</code> javascript objects.</td>
</tr>
</tbody>
</table>

### WorkflowModelManager - `getFinalExecutedActivityIdList()`

Gets the list of `wf_history.sys_ids` of all activities that successfully executed and were not rolled back or skipped up to the moment the function was called.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
var model = new WorkflowModelManager('ee3e0a053b101000dada82c09ccf3d7c');
model.getExecutedHistory();
var finals = model.getFinalExecutedActivityIdList();
gs.print(' EXECUTION PATH IDs --------------- : ' +
finals.length);
for ( var x = 0; x < finals.length; x++ ) {
    gs.print(finals[x] );
}

WorkflowModelManager - getNextByExecutedOrder(Object haRecord)

Retrieves the history activity that executed just after to the one provided in the argument.

The next status is based on the activity index reflecting the nearest prior activity in
time and not necessarily the nearest next wf_activity the provided activity transitioned
to. To get the nearest next activity that transitioned to this haRecord passed in, use
getNextByTransitionOrder( haRecord ).

Note: In the process of assembling the execution order array, blanks are left in the array
where history objects have been deleted. This is done to ensure the index in the array and
the activity_index of the object remain in synch. For that reason, all objects coming out
of the execution order array should be tested for nil() and not assumed to be the previous
record based on activity_index or array index value alone.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>haRecord</td>
<td>Object</td>
<td>ActivityHistoryRecord JavaScript object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The ActivityHistoryRecord JavaScript object requested.</td>
</tr>
</tbody>
</table>

WorkflowModelManager - getNextByExecutedOrderId(String haRecordSysId)

Retrieves the history activity that is cached by the wf_history.sys_id provided in the argument and
then calls into getNextByExecutedOrder() with the retrieved JavaScript object.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>haRecordSysId</td>
<td>String</td>
<td>The sys_id of the activity history (table wf_history).</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Array</td>
<td>Array of ActivityHistoryRecord JavaScript objects.</td>
</tr>
</tbody>
</table>

```javascript
var model = new WorkflowModelManager('7b3e01573b130000dada82c09ccf3dcf');
model.getExecutedHistory();
var current = model.getActivityHistoryRecordById('d6681d573b130000dada82c09ccf3d10');
current.debugDump();

var results = model.getNextByExecutedOrderId(current.sys_id);
results.debugDump();
```

**WorkflowModelManager - getNextByTransitionId(String haRecordSysId)**

Retrieves the history activity that executed just after to the one identified by the `sys_id` provided in the argument. This function retrieves the cached history record associated with the provided `wf_history.sys_id` and then calls `getNextByTransition`. The return values are based on which transitions came before the `haRecord` submitted and not necessarily the activities that executed just prior to the `haRecord` in time. To get the activity that executed prior to this activity in time, use `getNextByExecutedOrder`.

This function retrieves the cached history record associated with the provided `wf_history.sys_id` and then calls `getNextByTransition`. The return values are based on which transitions came before the `haRecord` submitted and not necessarily the activities that executed just prior to the `haRecord` in time. To get the activity that executed prior to this activity in time, use `getNextByExecutedOrder`.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>haRecordSysId</td>
<td>String</td>
<td><code>sys_id</code> of the activity history (table wf_history).</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Array</td>
<td>Array of ActivityHistoryRecord JavaScript objects.</td>
</tr>
</tbody>
</table>
var model = new WorkflowModelManager('7b3e01573b130000dada82c09ccf3dcf');
model.getExecutedHistory();
model.playBack();

var current = model.getActivityHistoryRecordById('d6681d573b130000dada82c09ccf3d10');
current.debugDump();

var results = model.getNextByTransitionId(current.sys_id);
gs.print('COMPLETED NEXT' + results.length);
for( var i = 0; i < results.length; i++){
    results[i].debugDump();
}

WorkflowModelManager - getPreviousByExecutedOrder(Object haRecord)

Retrieves the history activity that executed just previous to the one provided in the argument.

The previous status is based on the activity index reflecting the nearest prior activity in time and not necessarily the nearest prior activity with a valid transition to this activity. To get the nearest prior activity that transitioned to this haRecord passed in use getPreviousByTransition( haRecord).

Note: In the process of assembling the execution order array, blanks are left in the array where history objects have been deleted. This ensures that the index in the array and the activity_index of the object remain in sync. For that reason, all objects coming out of the execution order array should be tested for nil() and not assumed to be the previous record based on activity_index or array index value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>haRecord</td>
<td>Object</td>
<td>ActivityHistoryRecord JavaScript object</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>The ActivityHistoryRecord JavaScript object requested.</td>
</tr>
</tbody>
</table>

WorkflowModelManager - getPreviousByExecutedOrderId(String haRecordSysId)

Retrieves the history activity cached by the *wf_history.sys_id* provided in the argument, then calls getPreviousByExecutedOrder() with the retrieved JavaScript object.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>haRecordSysId</td>
<td>String</td>
<td>Workflow history system ID (wf_history.sys_id)</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivityHistoryRecord</td>
<td>The ActivityHistoryRecord JavaScript object requested.</td>
</tr>
</tbody>
</table>

```javascript
var model = new WorkflowModelManager('contextId');
model.getExecutedHistory();
var current = model.getActivityHistoryRecordById('wf_history.sys_id');
current.debugDump();

var results = model.getPreviousByExecutedOrderId(current.sys_id);
results.debugDump();
```

### WorkflowModelManager - getPreviousByTransition(Object haRecord)

Retrieves the history activities that executed just prior to the one provided in the argument.

The **next** status is based on the wf_activity.sys_id associated with the activity represented in the haRecord existing as a TO in a transition associated with any ActivityHistoryRecords that come before the haRecord in the execution sequence. (This differs from `getAllTransitionedIntoActivity`, which returns all TO transitions up to the given haRecord in the execution sequence.) The return value is a collection of ActivityHistoryRecords that identify the argument haRecord.wfaId as their TO activity. The return values are based on which transitions came before the haRecord submitted and not necessarily the activities that executed just prior to the haRecord in time. To get the activity that executed prior to this activity in time, use `getPreviousByExecutedOrder`.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>haRecord</td>
<td>Object</td>
<td>ActivityHistoryRecord JavaScript object.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Array</td>
<td>Array of ActivityHistoryRecord JavaScript objects requested.</td>
</tr>
</tbody>
</table>
WorkflowModelManager - getPreviousByTransitionId(String haRecordSysId)

Retrieves the history activities that executed just prior to the one provided in the argument. The next status is based on the \texttt{wf\_activity.sys\_id} associated with the activity represented in the haRecord existing as a TO in a transition associated with any ActivityHistoryRecords that come before the haRecord in the execution sequence. The return value is a collection of ActivityHistoryRecords that identify the argument haRecord.wf\_aid as their TO activity. The return values are based on which transitions came before the haRecord submitted and not necessarily the activities that executed just prior to the haRecord in time. To get the activity that executed prior to this activity in time use getPreviousByExecutedOrder.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>haRecordSysId</td>
<td>String</td>
<td>The sys_id of the activity history (table \texttt{wf_history}).</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Array of ActivityHistoryRecord JavaScript objects requested.</td>
</tr>
</tbody>
</table>

WorkflowModelManager - WorkflowModelManager(String contextId)

Creates a new WorkflowModelManager object (constructor) within the specified context.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextId</td>
<td>String</td>
<td>The ID for the context in which you want the WorkflowModelManager created.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivityHistoryRecord</td>
<td>The ActivityHistoryRecord JavaScript object requested.</td>
</tr>
</tbody>
</table>

```javascript
var model = new WorkflowModelManager('MyContextId');
```
WorkflowModelManagerAjax

Provides an AjaxProcessor wrapper for the WorkflowModelManager. This class derives from AbstractAjaxProcessor.

Use this class in the Workflow Debug context menus.

WorkflowModelManagerAjax - getExecutedHistory( )

Puts the execution history into the return parameter based on the query string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

WorkflowModelManagerAjax - getFinalExecutedActivityIdList( )

Puts the final execution activity list into the return parameter based on the query string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

WorkflowModelManagerAjax - getRolledBackActivityIdList( )

Puts the rollback activity list into the return parameter based on the query string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
WorkflowScheduler
This code is executed in business rule Set workflow scheduler script on table wf_workflow_schedule.

WorkflowScheduler - run()
Starts the workflow specified in current, which is a record in table wf_workflow_schedule.
This method is called from the Business Rule Set workflow scheduler script on table wf_workflow_schedule.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

WorkflowTimeline
Generates a view that shows a workflow context on the timeline.

WorkflowTimeline-getitems()
Generates the Ajax response for the timeline.
This method is called by the Prototype JavaScript Framework. See the AbstractTimelineSchedulePage script include for more details.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

XMLDocument2
XMLDocument2 is a JavaScript Object wrapper for parsing and extracting XML data from an XML string.
Use this JavaScript class to create an object from an XML string, usually a return value from a web-service invocation, or the XML payload of ECC Queue. Using the XMLDocument2 object in a JavaScript business rule lets you query values from the XML elements and attributes directly.
An XML string has a tree structure, and the parts of the structure are called nodes. An XMLDocument2 object deals with two node types, element, and document element. An element node is a node with a name and possibly attributes and child nodes. A document-element node is the root node of the XML tree. It is the only node without a parent node.

**Scoped XMLDocument2 - createElement(String name)**

Creates and adds an element node to the current node. The element name is the string passed in as a parameter. The new element has no text child nodes.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>The new element's name.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLNode</td>
<td>Current XML node.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>" +
    "  <one>" +
    "    <two att="xxx">abcd1234</two>" +
    "    <three att="yyy" boo="yah">1234abcd</three>" +
    "    <two>another</two>" +
    "  </one>" +
    "<number>1234</number>" +
"</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
xmlDoc.createElement("new2");
gs.info(xmlDoc);
```

**Output:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<test>
  <one>
    <two att="xxx">abcd1234</two>
    <three att="yyy" boo="yah">1234abcd</three>
    <two>another</two>
  </one>
  <number>1234</number>
  <new2></new2>
</test>
```

**Scoped XMLDocument2 - createElementWithTextValue(String name, String value)**

Creates and adds an element node with a text child node to the current node.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the element to add.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>Element's text value.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLNode</td>
<td>Current XML node.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>" +
"  <one>" +
"    <two att="xxx">abcd1234</two>" +
"    <three boo="yah" att="yyy">1234abcd</three>" +
"    <two>another</two>" +
"  </one>" +
"  <number>1234</number>" +
" </test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
xmlDoc.createElementWithTextValue("new", "test");
gs.info(xmlDoc);
```

```
Output:

<?xml version="1.0" encoding="UTF-8"?>
<test>
 <one>
    <two att="xxx">abcd1234</two>
    <three att="yyy" boo="yah">1234abcd</three>
    <two>another</two>
 </one>
 <number>1234</number>
 <new>test</new>
</test>
```

**Scoped XMLDocument2 - getDocumentElement()**

Gets the document element node of the XMLDocument2 object. The document element node is the root node.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLNode</td>
<td>The document element.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>
  <one>
    <two att="xxx">abcd1234</two>
    <three boo="yah" att="yyy">1234abcd</three>
  </one>
  <number>1234</number>
</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
//returns the root node of the document tree.
var rootNode = xmlDoc.getDocumentElement();
gs.info(rootNode.getTextContent());
```

Output:

```
abcd1234 1234abcd another 1234
```

Scoped XMLDocument2 - getFirstNode(String xPath)

Gets the first node in the specified XPATH.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xPath</td>
<td>String</td>
<td>The XPATH.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLNode</td>
<td>The first node.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>
  <one>
    <two att="xxx">abcd1234</two>
    <three boo="yah" att="yyy">1234abcd</three>
  </one>
  <number>1234</number>
</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var foo = xmlDoc.getFirstNode('/test/one/two');
gs.info(foo.getTextContent());
```
Output:

```
abcd1234
```

**Scoped XMLDocument2 - getNextNode(Object current)**

Gets the node after the specified node.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current</td>
<td>Object</td>
<td>The current node.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLNode</td>
<td>The next node.</td>
</tr>
</tbody>
</table>

```
var xmlString = "<test>" +
  "  <one>" +
    "    <two att="xxx">abcd1234</two>" +
    "    <three boo="yah" att="yyy">1234abcd</three>" +
  "  </one>" +
"  <number>1234</number>" +
"</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var foo = xmlDoc.getFirstNode('/test/one/two');
var foo2 = xmlDoc.getNextNode(foo);
gs.info(foo.getTextContent());
gs.info(foo2.getTextContent());
```

Output:

```
abcd1234
another
```

**Scoped XMLDocument2 - getNode(String xPath)**

Gets the node specified in the XPath.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xPath</td>
<td>String</td>
<td>XPath of the node to obtain.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLNode</td>
<td>Current XML node.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>
  <one>
    <two att="xxx">abcd1234</two>
    <three boo="yah" att="yyy">1234abcd</three>
  </one>
  <number>1234</number>
</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode("/test/one/two");
gs.info(node);
```

Output:

```
<two att="xxx">abcd1234</two>
```

Scoped XMLDocument2 - getNodeText(String xPath)

Gets all the text child nodes from the node referenced in the specified XPath.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xPath</td>
<td>String</td>
<td>XPath of the text to obtain.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Text children in the XPath.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>
  <one>
    <two att="xxx">abcd1234</two>
    <three boo="yah" att="yyy">1234abcd</three>
  </one>
  <number>1234</number>
</test>";
var xmlDoc = new XMLDocument2();
xmDoc.parseXML(xmlString);
gs.info(xmlDoc.getNodeText("//two"));`
Output: abcd1234

Scoped XMLDocument2 - parseXML(String xmlDoc)

Parses the XML string and loads it into the XMLDocument2 object.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmlDoc</td>
<td>String</td>
<td>The document to parse.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Flag that indicates whether the content was parsed.</td>
</tr>
</tbody>
</table>

This example parses the xmlString and loads it into the xmlDocument2 object.

```javascript
var xmlString = "<test>" + 
    "  <one>" + 
    "    <two att="xxx">abcd1234</two>" + 
    "    <three boo="yah" att="yyy">1234abcd</three>" + 
    "    <two>another</two>" + 
    "  </one>" + 
    "  <number>1234</number>" + 
    "</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var rootNode = xmlDoc.getDocumentElement();
```

Scoped XMLDocument2 - setCurrentElement(XMLNode element)

Makes the node passed in as a parameter the current node.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element</td>
<td>XMLNode</td>
<td>The element node to set as the current node.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
 Scoped XMLDocument2 - setNamespaceAware(Boolean aware)

When set to true, the XMLDocument2 object processes the document with XML namespaces.
If you don't set this, an XML document with namespaces won't be enumerated correctly, and an XPath search would fail.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aware</td>
<td>Boolean</td>
<td>When true, the XMLDocument2 object processes the document with XML namespaces.</td>
</tr>
</tbody>
</table>

Returns

Type | Description |
--- | -----------|
void |            |

 Scoped XMLDocument2 - toString()

Returns a string containing the XML.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A string containing the XML.</td>
</tr>
</tbody>
</table>
```
"  <one>
"    <two att="xxx">abcd1234</two>
"    <three att="yyy">1234abcd</three>
"  </one>
"  <number>1234</number>
</test>);

var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
gs.info(xmlDoc.toString());
```

Output - Line breaks were added to the output for readability.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<test>
  <one>
    <two att="xxx">abcd1234</two>
    <three att="yyy" boo="yah">1234abcd</three>
    <two>another</two>
  </one>
  <number>1234</number>
</test>
```

Scoped XMLDocument2 - XMLDocument2()

Creates an XMLDocument2 object.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scoped XMLDocument2 - XMLDocument2( GlideScriptableInputStream inputStream)

Creates an XMLDocument2 object from an attachment stream.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputStream</td>
<td>GlideScriptableInputStream</td>
<td>The input stream the XMLDocument2 object encapsulates.</td>
</tr>
</tbody>
</table>

XMLNode

The scoped XMLNode API allows you to query values from XML nodes. XMLNodes are extracted from XMLDocument2 objects, which contain XML strings.

There are no constructors for creating a stand alone instance of an XMLNode object. Instead, use the createElement() method of XMLDocument2, which adds a node to an existing document.

Scoped XMLNode - getAttribute(String attribute)

Gets the value of the attribute.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute</td>
<td>String</td>
<td>Name of the attribute.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The attribute's value.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>" +
"  <one>" +
"    <two att="xxx">abcd1234</two>" +
"    <three boo="yah" att="yyy">1234abcd</three>" +
"  <two>another</two>" +
"  </one>" +
"  <number>1234</number>" +
"</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode('//two');
gs.info(node.getAttribute('att'));
```

Output: xxx

### Scoped XMLNode - getAttributes()

Returns an object containing the node's attributes as properties with values.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Contains name-value pairs where the name is the attribute and the value is the attribute's value.</td>
</tr>
</tbody>
</table>

### Scoped XMLNode - getChildNodeIterator()

Gets a XMLNodeIterator object that can be used to walk through the list of child nodes.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### XMLNodeIterator

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLNodeIterator</td>
<td>The node iterator object.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>" +
    " <one>" +
    "    <two att="xxx">abcd1234</two>" +
    "    <three boo="yah" att="yyy">1234abcd</three>" +
    "    <two>another</two>" +
    " </one>" +
    " <number>1234</number>" +
"</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode('//one');
gs.info(iter.hasNext());
```

### Scoped XMLNode - getFirstChild()

**Gets the node's first child node.**

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLNode</td>
<td>The node's first child node.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>" +
    " <one>" +
    "    <two att="xxx">abcd1234</two>" +
    "    <three boo="yah" att="yyy">1234abcd</three>" +
    "    <two>another</two>" +
    " </one>" +
    " <number>1234</number>" +
"</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode('//one');
gs.info(node.getFirstChild());
```
Output:

```xml
<two att="xxx">abcd1234</two>
```

### Scoped XMLNode - getLastChild()

Gets the node's last child node.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLNode</td>
<td>The node's last child.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>" +
  "<one>" +
  "<two att="xxx">abcd1234</two>" +
  "<three boo="yah" att="yyy">1234abcd</three>" +
  "<two>another</two>" +
  "</one>" +
  "<number>1234</number>" +
  "</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode('//one');
gs.info(node.getLastChild());
```

Output:

```xml
<two>another</two>
```

### Scoped XMLNode - getNodeName()

Gets the node's name. A node's name is determined by the node type. A document-element node's name is #document. A text node's name is #text. An element node's name is the element's name.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The node's name.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>
  <one>
    <two att="xxx">abcd1234</two> +
    <three boo="yah" att="yyy">1234abcd</three>
  </one>
  <two>another</two> +
  <number>1234</number> +
</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode('//two');
gs.info(node.getNodeName());
```

Output: two

**Scoped XMLNode - getNodeValue()**

Gets the node's value. A node's value is determined by the node type. Element and document-element nodes return null.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The node's value.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>
  <one>
    <two att="xxx">abcd1234</two> +
    <three boo="yah" att="yyy">1234abcd</three>
  </one>
  <two>another</two> +
  <number>1234</number> +
</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode('//two');
gs.info(node.getNodeValue());
```
Output: null

Scoped XMLNode - getTextContent()

Gets the text content of the current node. The text content of a node consists of all the node's child text nodes.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The text content of the current node.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>" +
    "  <one>" +
    "    <two att="xxx">abcd1234</two>" +
    "    <three boo="yah" att="yyy">1234abcd</three>" +
    "    <two>another</two>" +
    "  </one>" +
    "  <number>1234</number>" +
"</test>";
var xmldoc = new XMLDocument2();
xmldoc.parseXML(xmlString);
var node = xmldoc.getNode('//one/two');
gs.info(node.getTextContent());
```

Output: abcd1234

Scoped XMLNode - hasAttribute(String attribute)

Determines if the node has the specified attribute.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute</td>
<td>String</td>
<td>The name of the attribute to check.</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the node has the attribute.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>
  <one>
    <two att="xxx">abcd1234</two>
    <three boo="yah" att="yyy">1234abcd</three>
  </one>
  <number>1234</number>
</test>";

var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode('//two');
gs.info(node.hasAttribute('att'));
```

Output: true

Scoped XMLNode - toString()

Returns the string value of the current node.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The string value of the current node.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>
  <one>
    <two att="xxx">abcd1234</two>
    <three boo="yah" att="yyy">1234abcd</three>
  </one>
  <number>1234</number>
</test>";

var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode('//one');
gs.info(node.toString());
```
Output: Line breaks were added to the output.

<one>
  <two att="xxx">abcd1234</two>
  <three att="yyy" boo="yah">1234abcd</three>
  <two>another</two>
</one>

**XMLNodeIterator**

The scoped XMLNodeIterator class allows you to iterate through a node of a XML document. There are no constructors for creating a stand alone instance of a XMLNodeIterator object. To create a XMLNodeIterator object use the `getChildNodeIterator()` method of the XMLNode object.

**Scoped XMLNodeIterator - hasNext()**

Returns true if the iteration has more elements.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>True if the iteration has more elements.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>" +
  "  <one>" +
  "    <two att="xxx">abcd1234</two>" +
  "    <three boo="yah" att="yyy">1234abcd</three>" +
  "    <two>another</two>" +
  "  </one>" +
  "  <number>1234</number>" +
"</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode('//one');
var iter = node.getChildNodeIterator();
gs.info(iter.hasNext());
```

**Scoped XMLNodeIterator - next()**

Gets the next element in the iteration. The returned element may be a #text node for the spaces/tabs if XML is “pretty formatted”.

```javascript
var xmlString = "<test>" +
  "  <one>" +
  "    <two att="xxx">abcd1234</two>" +
  "    <three boo="yah" att="yyy">1234abcd</three>" +
  "    <two>another</two>" +
  "  </one>" +
  "  <number>1234</number>" +
"</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode('//one');
var iter = node.getChildNodeIterator();
gs.info(iter.hasNext());
```
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLNode</td>
<td>The next element in the iteration.</td>
</tr>
</tbody>
</table>

```javascript
var xmlString = "<test>
  <one>
    <two att="xxx">abcd1234</two>
    <three boo="yah" att="yyy">1234abcd</three>
  </one>
  <number>1234</number>
</test>";
var xmlDoc = new XMLDocument2();
xmlDoc.parseXML(xmlString);
var node = xmlDoc.getNode('//one');
var iter = node.getChildNodeIterator();
while(iter.hasNext()) {
  var n = iter.next();
  gs.info('Node name: ' + n.getNodeName());
  gs.info('Node value: ' + n.getNodeValue());
}
```

Output:

Node name: #text
Node value:
Node name: two
Node value: null
Node name: #text
Node value:
Node name: three
Node value: null
Node name: #text
Node value:
Node name: two
Node value: null
Node name: #text
Node value:

**XMLUtilJS**

Provides XML utilities for JavaScript to be used with Discovery scripts.

Use this script include in any server-side discovery script where you need XML utilities.

Access these methods using the static variable `XMLUtilJS`. 
XMLUtilJS - escapeForXMLText(String text)
Provides escape text for a given string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>String</td>
<td>The text to format.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The formatted text.</td>
</tr>
</tbody>
</table>

XMLUtilJS - stringToValue(String str)
Converts a string to an XML value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>String</td>
<td>The string to convert</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The specified string converted to XML</td>
</tr>
</tbody>
</table>

XMLUtilJS - unescapeForXMLText(String text)
Provides un-escaped text for a given string.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>String</td>
<td>The text to clean up.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The cleaned up string.</td>
</tr>
</tbody>
</table>

XMLUtilJS - valueToString(String XMLvalue)
Converts an XML value to a string.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLvalue</td>
<td>String</td>
<td>The XML to convert</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The XML value converted to a string.</td>
</tr>
</tbody>
</table>

REST API reference

You can use REST interfaces to access data on your instance.

Agent Intelligence API

Use the Agent Intelligence API to predict a field value based on one or more input fields and a trained solution.

This API can only be used when the Agent Intelligence application is activated.

Agent Intelligence API - GET now/agent_intelligence/solution/{solution_name}/prediction

Predict an output field value using a specific solution.

URL format

Default URL: /api/now/agent_intelligence/solution/{solution_name}/prediction

The URL parameter.

- solution_name - The name of solution to use for predictions. For example, ml_incident_categorization.

Supported request parameters

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution definition input field key-value pair</td>
<td>Enter the name of the solution input field as a key-value pair. For example, enter the key, short_description and the value, Unable to connect to VPN.</td>
</tr>
</tbody>
</table>
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

**Request headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Response headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

**Status codes**

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response body

The API returns these JSON or XML elements in the response body to describe any prediction made.

**Elements returned in the response body**

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>outcome</td>
<td>The prediction output field value. For example, an incident categorization solution would return an incident category such as inquiry.</td>
</tr>
<tr>
<td>confidence</td>
<td>The estimated precision of the prediction as a percentage. For example, 53.84615375762915.</td>
</tr>
</tbody>
</table>
Sample cURL request

curl "https://instance.service-now.com/api/now/agent_intelligence/solution/ml_incident_categorization/prediction?short_description=unable%20to%20connect%20to%20VPN" \
--request GET \
--header "Accept:application/json" \
--user 'admin':'admin'

{
    "result": {
        "input": {
            "short_description": "unable to connect to VPN",
            "api": "api"
        },
        "output": {
            "outcome": "inquiry",
            "confidence": 53.84615375762915
        }
    }
}

Sample Python request

# Need to install requests package for python
# easy_install requests

import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/agent_intelligence/solution/ml_incident_categorization/prediction?short_description=unable%20to%20connect%20to%20VPN'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/json","Accept":"application/json"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers )

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

{
    "result": {
        "input": {

```
Aggregate API

The Aggregate API allows you to compute aggregate statistics about existing table and column data.

For Aggregate API requests, you must have read access for all records in the table you query. If an ACL prevents the requesting user from accessing any record in the table, the request returns a 403 Forbidden error.

Aggregate API - GET /now/stats/{tableName}

This method retrieves records for the specified table and performs aggregate functions on the returned values.

URL format

Versioned URL: /api/now/v1/stats/{tableName}
Default URL: /api/now/stats/{tableName}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_query</td>
<td>An encoded query.</td>
</tr>
<tr>
<td></td>
<td>For example: (sysparm_query=active=true) (sysparm_query=caller_id=javascript:gs.getUserID())</td>
</tr>
<tr>
<td></td>
<td>If part of the query is invalid, such as by specifying an invalid field name, the instance ignores the invalid part. It then returns rows using only the valid portion of the query. You can control this behavior using the property glide_invalid_query_returns_no_rows. Set this property to true to return no rows on an invalid query.</td>
</tr>
</tbody>
</table>

Note: This property controls the behavior of all queries across the instance, such as in lists, scripts (GlideRecord.query()), and web service APIs.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_group_by</td>
<td>Fields by which to group the returned data. You can specify multiple fields by separating each field with a comma, such as <code>sysparm_group_by=priority,state</code>.</td>
</tr>
<tr>
<td>sysparm_having</td>
<td>Additional query that enables you to filter the data based on an aggregate operation. The value for this parameter must follow the syntax <code>aggregate^field^operator^value</code>, such as <code>count^priority^&gt;^3</code> to obtain the number of records within the query results with a priority greater than 3. You can specify multiple queries by separating each with a comma, such as <code>count^state^=^1,avg^priority^&gt;^3</code>.</td>
</tr>
<tr>
<td>sysparm_&lt;aggregate&gt;_fields</td>
<td>List of fields on which to perform each aggregate operation. You can specify multiple fields by separating each with a comma. For example, to get the average values from the duration and priority fields, use <code>sysparm_avg_fields=duration,priority</code>. Note: Specify this parameter, the <code>sysparm_count</code> parameter, or both for your query to return meaningful results. If neither parameter is passed, no aggregate operation is performed.</td>
</tr>
<tr>
<td>sysparm_count</td>
<td>Flag that determines whether to return the number of records returned by the query. Note: Specify this parameter, the <code>sysparm_&lt;aggregate&gt;_fields</code> parameter, or both for your query to return meaningful results. If neither parameter is passed, no aggregate operation is performed.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| sysparm_display_value     | Data retrieval operation when grouping by reference or choice fields. Based on this value, the query returns either the display value, the actual value in the database, or both.  
  - **true** returns display values for all fields.  
  - **false** returns actual values from the database. If a value is not specified, this parameter defaults to false.  
  - **all** returns both actual and display values.  
  There is no preferred method for setting this parameter. However, specifying the display value may cause performance issues as they are not read from the database and may reference other fields and records. For more information on display values and actual values, see [Table API FAQs (KB0534905)](https://tableapisupport.servicenow.com/app/answers/detail/a_id/KB0534905). |
| sysparm_orderby           | List of values by which to order grouped results. You can specify an order using a field or an aggregate. For example, if you specify `sysparm_orderby=AVG^state`, groups of results with lower average state values are returned first. You can also order by COUNT to arrange groups of records by the number of records in each group.  
  When you specify an order, groups are ordered in ascending order by default. Use `^DESC` to sort in descending order, such as `sysparm_orderby=state^DESC`. |
| Key-Value Pairs           | An alternative to using the `sysparm_query` parameter. You can filter a query using key-value pairs where the key is the name of a field.  
  For example, instead of using the parameter `&sysparm_query=active=true`, you can use `&active=true`. You can use the display value when the field is a choice or reference type field, such as `&state=closed` instead of `&state=7`. To specify multiple key-value pairs, separate each with an ampersand, such as `&active=true&assigned_to=john.smith`. |

### Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see [Supported REST API headers](https://restapisupport.servicenow.com/app/answers/detail/a_id/KB0534905).
Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request completed successfully.</td>
</tr>
</tbody>
</table>

Available aggregate functions

You can specify which aggregate functions to perform by using either the sysparm_<aggregate>_fields parameter or sysparm_having=<aggregate>^field^operator^value parameter, substituting <aggregate> for one of these aggregate functions:

- avg
- sum
- min
- max

Sample cURL request

```
curl "https://instance.service-now.com/api/now/stats/incident?sysparm_avg_fields=reassignment_count%2Cbusiness_stc&sysparm_group_by=assignment_group" \
--request GET \
--header "Accept:application/json" \
--user 'admin':'admin'
```

```
{
  "result": [
    {
      "stats": {
        "avg": {
          "business_stc": "804162.7143",
          "reassignment_count": "1.0000"
        }
      }
    }
  ]
}
```
"groupby_fields": [
  {
    "value": "",
    "field": "assignment_group"
  }
],
"stats": {
  "avg": {
    "business_stc": "2037371.0000",
    "reassignment_count": "1.5000"
  }
},
"groupby_fields": [
  {
    "value": "287ee6fe9fe198100ada7950d0b1b73",
    "field": "assignment_group"
  }
],
"stats": {
  "avg": {
    "business_stc": "1821488.2857",
    "reassignment_count": "1.1111"
  }
},
"groupby_fields": [
  {
    "value": "8a5055c9c61122780043563ef53438e3",
    "field": "assignment_group"
  }
],
"stats": {
  "avg": {
    "business_stc": "1730322.0000",
    "reassignment_count": "1.2500"
  }
},
"groupby_fields": [
  {
    "value": "287ebd7da9fe198100f92cc8d1d2154e",
    "field": "assignment_group"
  }
],
"stats": {
  "avg": {
    "business_stc": "1564478.6250",
    "reassignment_count": "1.2500"
  }
}
Sample Python request

```python
#Need to install requests package for python
#easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/stats/incident?sysparm_avg_fields=reassignment_count%2Cbusiness_stc&sysparm_group_by=assignment_group'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type':'application/json','Accept':'application/json'}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)
```

```xml
<response>
  <result>
    <stats>
      <avg>
        <business_stc>804162.7143</business_stc>
        <reassignment_count>1.0000</reassignment_count>
      </avg>
    </stats>
    <groupby_fields>
      <field>assignment_group</field>
      <value />
    </groupby_fields>
  </result>
</response>
```
<groupby_fields>
</result>

<result>
<stats>
<avg>
<business_stc>2037371.0000</business_stc>
<reassignment_count>1.5000</reassignment_count>
</avg>
</stats>
<groupby_fields>
<field>assignment_group</field>
:value>287ee66ea9fe198100ada7950d0b1b73</value>
</groupby_fields>
</result>

<result>
<stats>
<avg>
<business_stc>1821488.2857</business_stc>
<reassignment_count>1.1111</reassignment_count>
</avg>
</stats>
<groupby_fields>
<field>assignment_group</field>
:value>8a5055c961122780043563ef53438e3</value>
</groupby_fields>
</result>

<result>
<stats>
<avg>
<business_stc>1730322.0000</business_stc>
<reassignment_count>1.2500</reassignment_count>
</avg>
</stats>
<groupby_fields>
<field>assignment_group</field>
:value>287ebd7da9fe198100f92cc8d1d215e</value>
</groupby_fields>
</result>

<result>
<stats>
<avg>
<business_stc>1564478.6250</business_stc>
<reassignment_count>1.2500</reassignment_count>
</avg>
</stats>
<groupby_fields>
<field>assignment_group</field>
:value>d625dccec0a8016700a222a0f7900d06</value>
</groupby_fields>
</result>

<result>
<stats>
<avg>
<business_stc>1512202.2500</business_stc>
<reassignment_count>1.1111</reassignment_count>
</avg>
</stats>
<groupby_fields>
<field>assignment_group</field>
:value>8a4d2e73c6112278017a6a4baf547aa7</value>
</groupby_fields>
</result>
Attachment API

The Attachment API allows you to upload and query file attachments. You can upload or retrieve a single file with each request. The Attachment API respects any system limitations on uploaded files, such as maximum file size and allowed attachment types. You can control these settings using the properties `com.glide.attachment.max_size`, 1024MB by default, and `glide.attachment.extensions`.

The following video provides more information on the Attachment API:

**Attachment API - DELETE /now/attachment/{sys_id}**

This method deletes the attachment with a specific sys_id value.

**URL format**

Versioned URL: /api/now/v1/attachment/{sys_id}
Default URL: /api/now/attachment/{sys_id}

**Supported request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>The sys_id value of the attachment to delete.</td>
</tr>
</tbody>
</table>

**Headers**

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

**Request headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Response headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see `REST response codes`.

### Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>Indicates the request ran successfully.</td>
</tr>
</tbody>
</table>

### Sample cURL request

```bash
curl "https://instance.service-now.com/api/now/
attachment/615ea769c0a8016601cf5f2367302f5" \
--request DELETE \
--header "Accept:application/json" \
--user 'admin':'admin'
```

### Sample Python request

```python
#Need to install requests package for python
#easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/
attachment/615ea769c0a8016601cf5f2367302f5'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/
xml","Accept":"application/xml"}  

# Do the HTTP request
response = requests.delete(url, auth=(user, pwd),
  headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:',
       response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
```
Attachment API - GET /now/attachment

This method gets the metadata for multiple attachments.

URL format

Versioned URL: api/now/v1/attachment
Default URL: api/now/attachment

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_query</td>
<td>An encoded query. Queries for the Attachment API are relative to the Attachments (sys_attachment) table. For example: (sysparm_query=file_name=attachment.doc) The encoded query provides support for order by. To sort responses based on certain fields, use the ORDERBY and ORDERBYDESC clauses in sysparm_query. For example, sysparm_query=ORDERBYfile_name^ORDERBYDESCtable_name orders the results in ascending order by name first, and then in descending order by table name. If part of the query is invalid, such as by specifying an invalid field name, the instance ignores the invalid part. It then returns rows using only the valid portion of the query. You can control this behavior using the property glide.invalid_query.returns_no_rows. Set this property to true to return no rows on an invalid query.</td>
</tr>
<tr>
<td>sysparm_limit</td>
<td>Limit to be applied on pagination. The default is 10000. Unusually large sysparm_limit values can impact system performance.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sysparm_offset</td>
<td>A number of records to exclude from the query. Use this parameter when you need to get more records than specified in sysparm_limit. For example, if sysparm_limit is set to 500, but there are additional records you want to query, you can specify a sysparm_offset value of 500 to get the second set of records.</td>
</tr>
</tbody>
</table>

**Headers**

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see [Supported REST API headers](#).

**Request headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

**Response headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>The content type of the response. For metadata requests, this is the content type of the metadata, not the content type of the attachment files.</td>
</tr>
<tr>
<td>Link</td>
<td>Links to download the attachments.</td>
</tr>
</tbody>
</table>

**Status codes**

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see [REST response codes](#).

**Status codes**

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the query ran successfully.</td>
</tr>
</tbody>
</table>

**Sample cURL request**

```bash
curl "https://instance.service-now.com/api/now/attachment?sysparm_limit=1" \
   --request GET \
   --header "Accept:application/json"
```
Sample Python request

```python
#Need to install requests package for python
#easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/attachment?sysparm_limit=1'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/xml","Accept":"application/xml"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
```
Attachment API - GET /now/attachment/{sys_id}

This method gets the metadata for the attachment file with a specific sys_id value.

URL format

Versioned URL: api/now/v1/attachment/<attachment record sys_id>
Default URL: api/now/attachment/<attachment record sys_id>

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>The sys_id of the attachment record you want to get metadata for.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.
### Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

### Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>The content type of the response. For metadata requests, this is the content type of the metadata, not the content type of the attachment files.</td>
</tr>
</tbody>
</table>

### Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see [REST response codes](#).

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the query ran successfully.</td>
</tr>
<tr>
<td>404</td>
<td>Indicates the specified attachment does not exist, or the current user cannot access it.</td>
</tr>
</tbody>
</table>

### Sample cURL request

```bash
curl "https://instance.service-now.com/api/now/attachment/615ea769c0a80166001cf5f2367302f5" \
--request GET \
--header "Accept:application/json" \
--user 'admin':'admin'
```

```json
{
    "result": {
        "table_sys_id": "5054b6f8c0a800060056addcf551ecf8",
        "size_bytes": "462",
        "download_link": "https://instance.service-now.com/api/now/attachment/615ea769c0a80166001cf5f2367302f5/file",
        "sys_updated_on": "2009-05-21 04:12:21",
        "sys_id": "615ea769c0a80166001cf5f2367302f5",
        "image_height": "",
        "sys_created_on": "2009-05-21 04:12:21",
        "file_name": "blocks.swf",
        "sys_created_by": "glide.maint",
        "compressed": "true",
        "average_image_color": "",
        "sys_updated_by": "glide.maint",
        "sys_tags": "",
        "table_name": "content_block_programmatic",
    }
}
```
Sample Python request

```python
# Need to install requests package for python
# easy_install requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/attachment/615ea769c0a80166001cf5f2367302f5'

# Eg. User name=\"admin\", Password=\"admin\" for this code sample.
user = \'admin\'
pwd = \'admin\'

# Set proper headers
headers = {\"Content-Type\":\"application/xml\",\"Accept\":\"application/xml\"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)
```

```xml
<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<response><result><table_sys_id>5054b6f8c0a800060056addcf551ecf8</table_sys_id><size_bytes>462</size_bytes><download_link>https://instance.service-now.com/api/now/attachment/615ea769c0a80166001cf5f2367302f5/file</download_link><sys_updated_on>2009-05-21 04:12:21</sys_updated_on><sys_id>615ea769c0a80166001cf5f2367302f5</sys_id><image_height /></result></response>
```
Attachment API - GET /now/attachment/{sys_id}/file
This method gets the binary file attachment with a specific sys_id value.

URL format

Versioned URL: api/now/v1/attachment/<attachment sys_id>/file
Default URL: api/now/attachment/<attachment sys_id>/file

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>The sys_id of the attachment record you want to get binary data from.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Attachment-Metadata</td>
<td>Metadata about the returned file, such as size, name, and file type.</td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.
<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the query ran successfully.</td>
</tr>
</tbody>
</table>

Sample cURL request

curl "https://instance.service-now.com/api/now/attachment/615ea769c0a80166001cf5f2367302f5/file" \
--request GET \
--header "Accept:*/*" \
--user 'admin':'admin'

Binary response not shown.

Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/attachment/615ea769c0a80166001cf5f2367302f5/file'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type':'application/xml', 'Accept':'*/*'}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

Binary response not shown.
```

Attachment API - POST /now/attachment/file

This method uploads a binary file specified in the request body as an attachment.
### URL format

Versioned URL: `/api/now/v1/attachment/file`
Default URL: `/api/now/attachment/file`

### Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_name (Required)</td>
<td>The name to give the attachment. This parameter is required to post an attachment.</td>
</tr>
<tr>
<td>table_name (Required)</td>
<td>The name of the table you want to attach the file to. This parameter is required to post an attachment.</td>
</tr>
<tr>
<td>table_sys_id (Required)</td>
<td>The sys_id of the record on the specified table that you want to attach the file to. This parameter is required to post an attachment.</td>
</tr>
<tr>
<td>encryption_context</td>
<td>The sys_id of an encryption context record. Specify this parameter to allow only users with the specified encryption context to access the attachment. If you do not specify this parameter, the attached file is not encrypted with any encryption context.</td>
</tr>
</tbody>
</table>

### Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see [Supported REST API headers](#).

#### Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>The content type of the file you want to attach. This header is mandatory to post file attachments.</td>
</tr>
</tbody>
</table>

#### Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>The URL of the new attachment.</td>
</tr>
</tbody>
</table>
Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Indicates the query ran successfully.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates that one or more mandatory parameters were missing from the request.</td>
</tr>
<tr>
<td>404</td>
<td>Indicates the record specified by the table_name and table_sys_id parameters does not exist or is not accessible by the current user.</td>
</tr>
</tbody>
</table>

Sample cURL request

```bash
curl "https://instance.service-now.com/api/now/attachment/file?table_name=incident&table_sys_id=d71f7935c0a8016700802b64c67c11c6&file_name=Issue_screenshot" \
--request POST \
--header "Accept:application/json" \
--user 'admin':'admin' \
--header "Content-Type: image/jpeg" \
--data-binary "@ location of the file on file system"
```

```json
{
    "result": {
        "table_sys_id": "d71f7935c0a8016700802b64c67c11c6",
        "size_bytes": "36597",
        "download_link": "https://instance.service-now.com/api/now/attachment/6ea10fe64f411200adf9f8e18110c739/file",
        "sys_updated_on": "2016-01-22 15:14:07",
        "sys_id": "6ea10fe64f411200adf9f8e18110c739",
        "image_height": "",
        "sys_created_on": "2016-01-22 15:14:07",
        "file_name": "Issue_screenshot",
        "sys_created_by": "admin",
        "compressed": "true",
        "average_image_color": "",
        "sys_updated_by": "admin",
        "sys_tags": "",
        "table_name": "incident",
        "image_width": "",
        "sys_mod_count": "0",
        "content_type": "image/jpeg",
        "size_compressed": "25130"
    }
}
```
Sample Python request

```python
# Need to install requests package for python
# easy_install requests

import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/attachment/file?'
table_name=incident&table_sys_id=d71f7935c0a8016700802b64c67c11c6&file_name=Issue_screenshot.jpg

# Specify the file To send. When specifying files to send make sure you specify the path to the file, in this example the file was located in the same directory as the python script being executed.
data = open('Issue_screenshot.jpg', 'rb').read()

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"image/jpeg","Accept":"application/json"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd), headers=headers, data=data)

# Check for HTTP codes other than 201
if response.status_code != 201:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

{
    "result": {
        "table_sys_id": "d71f7935c0a8016700802b64c67c11c6",
        "size_bytes": "36597",
        "download_link": "https://instance.service-now.com/api/now/attachment/6ea10fe64f411200adf9f8e18110c739/file",
        "sys_updated_on": "2016-01-22 15:14:07",
        "sys_id": "6ea10fe64f411200adf9f8e18110c739",
        "image_height": "",
        "sys_created_on": "2016-01-22 15:14:07",
        "file_name": "Issue_screenshot.jpg",
        "sys_created_by": "admin",
        "compressed": "true",
        "average_image_color": "",
        "sys_updated_by": "admin",
        "sys_tags": "",
        "table_name": "incident",
        "image_width": "",
        "sys_mod_count": "0",
        "content_type": "image/jpeg",
        "size_compressed": "25130"
    }
}
Attachment API - POST /now/attachment/upload

This method uploads a multipart file attachment.

URL format

Versioned URL: /api/now/v1/attachment/upload
Default URL: /api/now/attachment/upload

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>The multipart POST method does not accept any parameters. The table name and record sys_id values must be specified within the message body. See the POST multipart sample for an example of a multipart message.</td>
</tr>
</tbody>
</table>

Important: When using multipart POST, ensure the file content is contained in the final part of the message only. Earlier parts should contain only metadata such as table name and record sys_id.

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>The content type of the request. Set this value to multipart/form-data when using the multipart POST method.</td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>The URL of the new attachment.</td>
</tr>
</tbody>
</table>
Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see *REST response codes*.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Indicates the query ran successfully.</td>
</tr>
</tbody>
</table>

POST multipart mandatory values

When sending a multipart POST request to upload a file attachment, include attachment data in the message body, not in the URL parameters. You must specify these values in the message body:

### Mandatory values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the table to which you want to attach the file.</td>
</tr>
<tr>
<td>table_sys_id</td>
<td>The sys_id of the record on the specified table to which you want to attach the file.</td>
</tr>
<tr>
<td>Content-Type</td>
<td>The Content-Type of the file, included in the message body for multipart uploads.</td>
</tr>
</tbody>
</table>

**Note:** The Content-Type must be defined within the file portion of the POST message, not within the form data. See the sample POST multipart message for an example of a multipart message.

Sample cURL request

```bash
curl "https://instance.service-now.com/api/now/attachment/upload" \
  --request POST \
  --header "Accept:application/json" \
  --user 'username':'password'\n  --header "Content-Type:multipart/form-data" \
  -F 'table_name=incident' -F 'table_sys_id=d71f7935c0a8016700802b64c67c11c6' -F 'uploadFile=@ location of the file on file system'
```

```json
{
  "result": {
    "table_sys_id": "d71f7935c0a8016700802b64c67c11c6",
    "size_bytes": "36597",
    "download_link": "https://instance.service-now.com/api/now/attachment/994adbc64f511200adf9f8e18110c796/file",
  }
}
```
Sample Python request

```python
# This example uses the Python Requests Library and you will
# need to install requests package for python
# Documentation can be found at http://docs.python-requests.org/
# en/master/user/quickstart/
import requests
import pprint
import json

# Specify the Endpoint URL replacing instance with your
# ServiceNow Instance Name
url = 'https://instance.service-now.com/api/now/attachment/
    upload'

# Specify Parameters for File Being Uploaded, the table_name and
# table_sys_id should be replaced with values that make
# sense for your use case
payload = {'table_name':'incident',
           'table_sys_id':'81f8915b6ba2002892416bf961971'}

# Specify Files To Send and Content Type. When specifying files
# to send make sure you specify the path to the file, in
# this example the file was located in the same directory as the
# python script being executed.
# it is important to specify the correct file type
files = {'file': ('issue_screenshot.JPG',
                 open('issue_screenshot.JPG', 'rb'), 'image/jpg',
                 {'Expires': '0'})}

# Eg. User name="username", Password="password" for this code
# sample. This will be sent across as basic authentication
user = 'username'
pwd = 'password'

# Set proper headers
headers = {'Accept': '*/*'}

# Send the HTTP request
response = requests.post(url, auth=(user, pwd), headers=headers,
                          files=files, data=payload)

# Check for HTTP codes other than 201
```
if response.status_code != 201:
    print('Status:', response.status_code, 'Headers:',
    response.headers, 'Error Response:', response.json())
    exit()

# Print Response Details
print 'Response Status Code:', response.status_code
print ''
print('Response Payload:')
print json.dumps(response.json(), indent=4)

Response Status Code: 201
Response Payload:
{
    "result": {
        "sys_tags": "",
        "sys_updated_by": "admin",
        "content_type": "image/jpg",
        "sys_created_by": "admin",
        "file_name": "issue_screenshot.JPG",
        "sys_updated_on": "2017-01-05 10:47:09",
        "sys_created_on": "2017-01-05 10:47:09",
        "image_width": "",
        "image_height": "",
        "sys_mod_count": "0",
        "table_name": "incident",
        "sys_id": "96679f724f84320025e874828110c7bd",
        "download_link": "https://instance.service-now.com/api/now/attachment/96679f724f84320025e874828110c7bd/file",
        "average_image_color": "",
        "size_bytes": "197484",
        "table_sys_id": "81f8915bd6ba20028927416bf961971",
        "size_compressed": "197005",
        "compressed": "true"
    }
}

**CI Lifecycle Management API**

Provides the ability to manipulate CI operational states and apply CI actions.

The CI Lifecycle Management API interfaces adhere to restrictions and allowances specified by not allowed CI actions, compatible CI actions, and not allowed operational transitions. If an interface attempts a restricted operation, the operation is blocked, an error is logged, and a task is created if appropriate.

**CI Lifecycle Management API - DELETE /now/cilifecyclemgmt/actions**

Removes a CI action for a list of CIs.

**URL format**

Versioned URL: api/now/v1/cilifecyclemgmt/actions
Default URL: api/now/cilifecyclemgmt/actions
Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestorid</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator (POST) interface.</td>
</tr>
<tr>
<td>sysIds</td>
<td>A comma separated list of CI sys_ids</td>
</tr>
<tr>
<td>actionName</td>
<td>The CI action name</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

Response payload

A JSON formatted string with name-value pairs for
- result - true if the action is removed for all CIs in the list; otherwise, false.
- errors - list of errors.
CI Lifecycle Management API - DELETE /now/cilifecyclemgmt/operators/{req_id}
Unregister an operator for non-workflow users.

URL format

Versioned URL: api/now/v1/cilifecyclemgmt/operators/{req_id}
Default URL: api/now/cilifecyclemgmt/operators/{req_id}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>req_id</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator (POST) interface.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>
Response payload

A JSON formatted string with name-value pairs for
- result - true if successfully unregistered.
- errors - list of errors.

CI Lifecycle Management API - GET /now/cilifecyclemgmt/actions/{sys_id}

Returns a list of active CI actions for the specified CI.

URL format

Versioned URL: api/now/v1/cilifecyclemgmt/actions/{sys_id}
Default URL: api/now/cilifecyclemgmt/actions/{sys_id}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>The sys_id of the CI</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.
Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

Response payload

A JSON formatted string with name-value pairs for
- ciActions - a comma separated list of active CI actions for the CI, or NO_ACTIVE_ACTION.
- errors - list of errors.

CI Lifecycle Management API - GET /now/cilifecyclemgmt/compatActions

Determines if the two specified actions are compatible.

URL format

Versioned URL: api/now/v1/cilifecyclemgmt/compatActions
Default URL: api/now/cilifecyclemgmt/compatActions

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionName</td>
<td>A CI action name</td>
</tr>
<tr>
<td>otherActionName</td>
<td>A CI action name</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>
Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

Response payload

Returns true if the two specified CI actions are compatible; otherwise false.

CI Lifecycle Management API - GET /now/cilifecyclemgmt/leases/{sys_id}/expired

Determines if the lease has expired for the requestor of a specified CI Action.

URL format

Versioned URL: api/now/v1/cilifecyclemgmt/leases/{sys_id}/expired
Default URL: api/now/cilifecyclemgmt/leases/{sys_id}/expired

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>The CI's sys_id.</td>
</tr>
<tr>
<td>requestorid</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator (POST) interface.</td>
</tr>
<tr>
<td>actionName</td>
<td>The CI action name.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers
Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

Response payload

A JSON formatted string with name-value pairs for
- result - true if the user’s lease has expired; otherwise, false.
- errors - list of errors.

CI Lifecycle Management API - GET /now/cilifecyclemgmt/notAllowedAction

For a type of CI, determine if a CI action is not allowed for an operational state.

URL format

Versioned URL: api/now/v1/cilifecyclemgmt/notAllowedAction
Default URL: api/now/cilifecyclemgmt/notAllowedAction

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciClass</td>
<td>The CI type</td>
</tr>
<tr>
<td>opsLabel</td>
<td>The operational state</td>
</tr>
<tr>
<td>actionName</td>
<td>The CI action name</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.
Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

Response payload

Returns true if the action is not allowed on the specified CI type in the specified operational state.

CI Lifecycle Management API - GET /now/cilifecyclemgmt/notAllowedOpsTransition

Brief introduction to the topic that appears on overview pages and in link previews.

URL format

Versioned URL: api/now/v1/cilifecyclemgmt/notAllowedOpsTransition
Default URL: api/now/cilifecyclemgmt/notAllowedOpsTransition

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciClass</td>
<td>The CI type</td>
</tr>
<tr>
<td>opsLabel</td>
<td>The label of the beginning operational state.</td>
</tr>
<tr>
<td>transitionOpsLabel</td>
<td>The label of the ending operational state.</td>
</tr>
</tbody>
</table>
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

### Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

### Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

### Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

### Response payload

Returns true if specified operational state transition is not allowed on the specified CI type.

**CI Lifecycle Management API - GET /now/cilifecyclemgmt/requestors/{req_id}/valid**

Determine if the specified requestor is a valid active workflow user or a registered user.

### URL format

- **Versioned URL**: api/now/v1/cilifecyclemgmt/requestors/{req_id}/valid
- **Default URL**: api/now/cilifecyclemgmt/requestors/{req_id}/valid
Supported request parameters

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>req_id</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator (POST) interface.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

Response payload

A JSON formatted string with name-value pairs for
- result - true if the req_id is valid.
- errors - list of errors.

CI Lifecycle Management API - GET /now/cilifecyclemgmt/statuses/{sys_id}

Returns the CI's operational state.
URL format

Versioned URL: api/now/v1/cilifecyclemgmt/statuses/{sys_id}
Default URL: api/now/cilifecyclemgmt/statuses/{sys_id}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>The sys_id of the CI</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

<table>
<thead>
<tr>
<th>Request headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status code</td>
</tr>
<tr>
<td>200</td>
</tr>
</tbody>
</table>

Response payload

A JSON formatted string with name-value pairs for
- operationalStates - one of the string choice values of the operational_status field, or UNKNOWN.
- errors - list of errors.
CI Lifecycle Management API - PATCH /now/cilifecyclemgmt/leases/{sys_id}

Extend the CI-action-lease time for the registered user. If the previous lease has expired, the new lease time starts now.

URL format

Versioned URL: api/now/v1/cilifecyclemgmt/leases/{sys_id}
Default URL: api/now/cilifecyclemgmt/leases/{sys_id}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>The CI's sys_id.</td>
</tr>
<tr>
<td>requestorid</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator (POST) interface.</td>
</tr>
<tr>
<td>actionName</td>
<td>The CI action name.</td>
</tr>
<tr>
<td>leaseTime</td>
<td>Time duration for which the lease is valid for specified CI Action. In the format HH:MM:SS.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.
### Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

### Response payload

A JSON formatted string with name-value pairs for
- result - true if the lease time has been set.
- errors - list of errors.

### CI Lifecycle Management API - POST /now/cilifecyclemgmt/actions

Add an action to a list of CIs.

### URL format

**Versioned URL:** api/now/v1/cilifecyclemgmt/actions

**Default URL:** api/now/cilifecyclemgmt/actions

### Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestorId</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator (POST) interface.</td>
</tr>
<tr>
<td>sysIds</td>
<td>A list comma separated list of CI sys_ids.</td>
</tr>
<tr>
<td>actionName</td>
<td>The CI action name.</td>
</tr>
<tr>
<td>oldActionNames</td>
<td>(Optional) A comma separated list of old CI actions that all CIs should be in.</td>
</tr>
<tr>
<td>leaseTime</td>
<td>(Optional) Time duration for which the lease is valid for specified CI Action. In the format HH:MM:SS.</td>
</tr>
</tbody>
</table>

### Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see [Supported REST API headers](#).
Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

Response payload

A JSON formatted string with name-value pairs for

- result - true if the action is set for all CIs in the list; otherwise, false.
- errors - list of errors.

CI Lifecycle Management API - POST /now/cilifecyclemgmt/operators

Register an operator for a non-workflow user.

URL format

- Versioned URL: api/now/v1/cilifecyclemgmt/operators
- Default URL: api/now/cilifecyclemgmt/operators

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
**Headers**

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see [Supported REST API headers](#).

### Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

### Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

**Status codes**

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see [REST response codes](#).

### Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

**Response payload**

A JSON formatted string with name-value pairs for

- `requestorid` - Registered user GUID that is used to set CI Action/operational states.
- `result` - true if successfully registered.
- `errors` - list of errors.

**CI Lifecycle Management API - POST /now/cilifecyclemgmt/statuses**

Set the operational state for list of CIs.

**URL format**

- Versioned URL: api/now/v1/cilifecyclemgmt/statuses
- Default URL: api/now/cilifecyclemgmt/statuses
Supported request parameters

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestorId</td>
<td>A sys_id of a workflow context, or the GUID returned from the registerOperator() method.</td>
</tr>
<tr>
<td>sysIds</td>
<td>A comma separated list of CI sys_ids.</td>
</tr>
<tr>
<td>opsLabel</td>
<td>This is the string label of an operational_status choice.</td>
</tr>
<tr>
<td>oldOpsLabels</td>
<td>(Optional) A comma separated list of old CI states that all CIs should be in.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

Response payload

A JSON formatted string with name-value pairs for

- result - true if the state is set for all CIs in the list; otherwise, false.
- errors - list of errors.

**MetricBase Time Series API**

Use the MetricBase Time Series API to insert data into, retrieve information from, and to run transforms against a MetricBase database.

These APIs can only be used when the MetricBase plugin (com.snc.clotho) is installed and activated.

Role required to write to this API: clotho_rest_put

The examples in this section were created using data in the MetricBase Demo plugin (com.snc.clotho.demo)

**MetricBase Time Series API - GET now/v1/clotho/table/{table}/{subject}/{metric}**

Retrieve time series data from the MetricBase database.

**URL format**

Versioned URL: api/now/{version}/clotho/table/{table}/{subject}/{metric}

The URL parameters.

- **table**: Name of the table containing the GlideRecord associated with this series.
- **subject**: The sys_id of the GlideRecord associated with this series.
- **metric**: The field name of the metric.

**Supported request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_start</td>
<td>The start time of the evaluation period in UTC. The value must be in ISO date format up to seconds. For example, 2017-03-20T17:04:55.</td>
</tr>
<tr>
<td>sysparm_end</td>
<td>The end time of the evaluation period in UTC. The value must be in ISO date format up to seconds. For example, 2017-03-20T17:04:55.</td>
</tr>
<tr>
<td>sysparm_display_value</td>
<td>Optional. If true, the result data is labeled with the subject record display value.</td>
</tr>
</tbody>
</table>

**Headers**

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.
### Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

### Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

### Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see [REST response codes](#).

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

### Response body

The API returns a JSON object or an array of JSON objects. The format of the JSON object is in this form, where:

- **seriesRef**: defines the table, record, and metric.
- **subject**: the sys_id of the specific record in the table.
- **table**: the name of the table containing the record.
- **metric**: the metric field name.
- **values**: the individual data points, consisting of a time and value.

```json
{"seriesRef": {"subject": "xxx", "table": "yyy", "metric": "zzz"},
"values": [{"timestamp": "2016-10-14T10:30:00", "value": 1.23},
{"timestamp": "2016-10-14T10:31:00", "value": 6.45}]
```

### Sample cURL request

```
curl "https://<instance>.service-now.com/api/now/v1/clotho/table/mb_demo_drone/626b051787333200a328c5b836cb0b99/mb_demo_mt_altitude?sysparm_start=2019-03-20T17%3A04%3A55&sysparm_end=2019-03-20T17%3A09%3A55"
--request GET \
--header "Accept:application/json"
--user 'admin':'admin'
```

```json
{
"seriesRef": {
"subject": "626b051787333200a328c5b836cb0b99",
"table": "mb_demo_drone",
```
Sample Python request

# Need to install requests package for python
# easy_install requests

import requests

# Set the request parameters
url = 'https://<instance>.service-now.com/api/now/v1/clotho/table/mb_demo_drone/626b051787333200a328c5b836cb0b99/mb_demo_mt_altitude?sysparm_start=2019-03-20T17%3A04%3A55&sysparm_end=2019-03-20T17%3A09%3A55'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/json",
           "Accept":"application/json"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers )

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

{
   "seriesRef": {
      "subject": "626b051787333200a328c5b836cb0b99",
      "table": "mb_demo_drone",
      "metric": "mb_demo_mt_altitude"
   },
   "label": "626b051787333200a328c5b836cb0b99:mb_demo_drone|mb_demo_mt_altitude",
   "values": [
      {
         "timestamp": "2019-03-20T17:05:00Z",
         "value": 83.150185
      },
      {
         "timestamp": "2019-03-20T17:06:00Z",
         "value": 83.46074
      },
      {
         "timestamp": "2019-03-20T17:07:00Z",
         "value": 83.83104
      },
      {
         "timestamp": "2019-03-20T17:08:00Z",
         "value": 84.260635
      },
      {
         "timestamp": "2019-03-20T17:09:00Z",
         "value": 84.749
      }
   ]
}

MetricBase Time Series API - GET now/v1/clotho/transform/{table}/{metric}
Transforms selected data.

URL format

Versioned URL: /api/now/{version}/clotho/transform/{table}/{metric}

URL parameters:

- table: Name of the table containing the GlideRecord associated with this series.
- metric: The field name of the metric.

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_query</td>
<td>An encoded query string for finding the subject</td>
</tr>
<tr>
<td></td>
<td>records.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Parameter | Description
--- | ---
`sysparm_start` | The start time of the evaluation period in UTC. The value must be in ISO date format up to seconds. For example, 2017-03-20T17:04:55.
`sysparm_end` | The end time of the evaluation period in UTC. The value must be in ISO date format up to seconds. For example, 2017-03-20T17:04:55.
`sysparm_transforms` | Optional. A comma separated list of transforms. Supported transforms are avg, sum, add, mul, resample, top, label.
`sysparm_subject_limit` | Optional. Limit the size of the subject query result. The default is 10,000.
`sysparm_display_value` | Optional. When true, the result data will be labeled with the subject record display value if no other label is specified in the transform. The default is false.

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
Response body

The API returns a JSON object or an array of JSON objects. The format of the JSON object is in this form, where

- **seriesRef**: defines the table, record, and metric.
- **subject**: the sys_id of the specific record in the table.
- **table**: the name of the table containing the record.
- **metric**: the metric field name.
- **values**: the individual data points, consisting of a time and value.

```
{"seriesRef":{"subject":"xxx","table":"yyy","metric":"zzz"},
"values": [{"timestamp":"2016-10-14T10:30:00","value":1.23},
{"timestamp":"2016-10-14T10:31:00","value":6.45}]
```

Sample cURL request

```
curl "https://<instance>.service-now.com/api/now/v1/clotho/transform/mb_demo_drone/mb_demo_mt_speed?
sysparm_query=model%3DKingfisher%20Phantom&
sysparm_start=2019-03-25T17%3A04%3A55&
sysparm_end=2019-03-25T17%3A05%3A10" 
--request GET 
--header "Accept:application/json" 
--user 'admin':'admin'
```

```
[
  {
    "seriesRef": {
      "subject": "2a6b051787333200a328c5b836cb0b92",
      "table": "mb_demo_drone",
      "metric": "mb_demo_mt_speed"
    },
    "label": "2a6b051787333200a328c5b836cb0b92:mb_demo_drone|mb_demo_mt_speed",
    "values": [
      {
        "timestamp": "2019-03-25T17:05:00Z",
        "value": 33.67892
      }
    ]
  },
  {
    "seriesRef": {
      "subject": "666b051787333200a328c5b836cb0b92",
      "table": "mb_demo_drone",
      "metric": "mb_demo_mt_speed"
    },
    "label": "666b051787333200a328c5b836cb0b92:mb_demo_drone|mb_demo_mt_speed",
    "values": [
      {
        "timestamp": "2019-03-25T17:05:00Z",
        "value": 41.94985
      }
    ]
  }
]```
Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://<instance>.service-now.com/api/now/{v1}/
clotho/transform/mb_demo_drone/mb_demo_mt_speed?
sysparm_query=model%3DKingfisher%20Phantom&
sysparm_start=2019-03-25T17%3A04%3A55&
sysparm_end=2019-03-25T17%3A05%3A10'

# Eg. User name="admin", Password="admin" for code sample.
user = 'admin'
pwd = 'admin'
```
# Set proper headers
headers = {"Content-Type":"application/json",
           "Accept":"application/json"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd),
                        headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:',
          response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

[
  {
    "seriesRef": {
    "subject": "2a6b051787333200a328c5b836cb0b92",
    "table": "mb_demo_drone",
    "metric": "mb_demo_mt_speed"
    },
    "label": "2a6b051787333200a328c5b836cb0b92:mb_demo_drone
            |mb_demo_mt_speed",
    "values": [
    {
      "timestamp": "2019-03-25T17:05:00Z",
      "value": 33.67892
    }
    ]
  },
  ...
  {
    "seriesRef": {
    "subject": "ee6b051787333200a328c5b836cb0b91",
    "table": "mb_demo_drone",
    "metric": "mb_demo_mt_speed"
    },
    "label": "ee6b051787333200a328c5b836cb0b91:mb_demo_drone
            |mb_demo_mt_speed",
    "values": [
    {
      "timestamp": "2019-03-25T17:05:00Z",
      "value": 44.170887
    }
    ]
  }
]
URL format

Versioned URL: api/now/{version}/clotho/put

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_ignore_unknown_series</td>
<td>Optional. If true, ignore unknown series and continue the transaction without returning an error. The default is true.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

**Request headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Response headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Request body

The request body takes a JSON object or an array of JSON objects. The format of the JSON object is:

- seriesRef — Defines the table, record, and metric.
• subject — sys_id of the specific record in the table.
• table — Name of the table containing the record.
• metric — Metric field name.
• values — An array of values, each consisting of a:
  • timestamp — UTC timestamp in the ISO format: YYYY-MM-ddTHH:mm:dd. The ending, ‘Z’, which denotes the UTC time zone in an ISO-formatted timestamp is optional.
  • value — Float

The response is a status code, for example, 200.

Sample cURL request

```bash
curl "https://<instance>.service-now.com/api/now/{v1}/clotho/put" \
--request POST \n--header "Accept:application/json" \n--header "Content-Type:application/json" \n--data "{
  "seriesRef": {
    "subject": "3D666b05178733320a328c5b836cb0b92", 
    "table": "mb_demo_drone", 
    "metric": "mb_demo_mt_altitude"
  }, 
  "values": [
    {
      "timestamp": "2019-03-21T17:05:00Z", 
      "value": 0.150185
    }, 
    {
      "timestamp": "2019-03-21T17:06:00Z", 
      "value": 0.46074
    }, 
    {
      "timestamp": "2019-03-21T17:07:00Z", 
      "value": 0.83104
    }, 
    {
      "timestamp": "2019-03-21T17:08:00Z", 
      "value": 1.260635
    }, 
    {
      "timestamp": "2019-03-21T17:09:00Z", 
      "value": 1.749
    }
  ]
}"
--user 'admin':'admin'
```

```json
{
  "result": {
    "message": "ok"
  }
}
```

Sample Python request

```python
#Need to install requests package for python
```
#easy_install requests

import requests

# Set the request parameters
url = 'https://<instance>.service-now.com/api/now/{v1}/clotho/put'

# Eg. User name="admin", Password="admin" for this sample
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/json",
           "Accept":"application/xml"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd),
                          headers=headers, data="{
                          "seriesRef": {,
                            "subject": "3D666b051787333200a328c5b836cb0b92",
                            "table": "mb_demo_drone",
                            "metric": "mb_demo_mt_altitude"
                          },
                          "values": [
                          {
                            "timestamp": "2019-03-21T17:05:00Z",
                            "value": 0.150185
                          },
                          {
                            "timestamp": "2019-03-21T17:06:00Z",
                            "value": 0.46074
                          },
                          {
                            "timestamp": "2019-03-21T17:07:00Z",
                            "value": 0.83104
                          },
                          {
                            "timestamp": "2019-03-21T17:08:00Z",
                            "value": 1.260635
                          },
                          {
                            "timestamp": "2019-03-21T17:09:00Z",
                            "value": 1.749
                          }
                        ]
                      }")

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:',
          response.headers, 'Error Response:', response.json())
    exit()

# Decode JSON response into a dictionary and use the data
data = response.json()
print(data)

<?xml version="1.0" encoding="UTF-8"?>
<response>
  <result>
    <message>ok</message>
  </result>
**CMDB instance API**

Perform create, read, update, and delete operations on existing CMDB tables. Only one record can be inserted, updated, or deleted at a time.

These APIs require that the user have the ITIL role.

**CMDB instance API - DELETE - /now/cmdb/instance/{classname}/{sys_id}/relation/{rel_sys_id}**

Delete the relation for the specified CI.

**URL format**

Versioned URL: `api/now/v1/cmdb/instance/{className}/{sys_id}/relation/{rel_sys_id}`

Default URL: `api/now/cmdb/instance/{className}/{sys_id}/relation/{rel_sys_id}`

The path parameters.
- `className` - The CMDB class name.
- `sys_id` - The record sys_id.
- `rel_sys_id` - The sys_id of the relation.

**Supported request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_fields</td>
<td>A comma-separated list of fields to return in the response</td>
</tr>
<tr>
<td>sysparm_relation_limit</td>
<td>The maximum number of relations to return. Default is 1000.</td>
</tr>
<tr>
<td>sysparm_relation_offset</td>
<td>A number of records to exclude from the relations query. Default is 0.</td>
</tr>
</tbody>
</table>

**Headers**

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see [Supported REST API headers](#).
Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

CMDB instance API - GET - /now/cmdb/instance/{classname}

Query records for a CMDB class.

URL format

Versioned URL: api/now/v1/cmdb/instance/{className}
Default URL: api/now/cmdb/instance/{className}

The path parameter.
- className - The CMDB class name.

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_query</td>
<td>An encoded query string used to filter the results</td>
</tr>
<tr>
<td>sysparm_limit</td>
<td>The maximum number of results to return per page. Default is 1000.</td>
</tr>
<tr>
<td>sysparm_offset</td>
<td>A number of records to exclude from the query. Default is 0.</td>
</tr>
</tbody>
</table>
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

**Request headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Response headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

**Status codes**

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response body

The API returns these JSON or XML elements in the response body.

```json
{"result": [{
  "sys_id": "3a290cc60a0a0bb400000bdb386af1cf",
  "name": "PS LinuxApp01"},
  {  
  "sys_id": "3a5dd3db0a8ce0100655f1ec66ed42c",
  "name": "PS LinuxApp02"}]}  
```

**CMDB instance API - GET -/now/cmdb/instance/{classname}/{sys_id}**

Query attributes and relationship information for a specific record.

**URL format**

Versioned URL: api/now/v1/cmdb/instance/{className}/{sys_id}

Default URL: api/now/cmdb/instance/{className}/{sys_id}

The path parameters.
- className - The CMDB class name.
- sys_id - The record sys_id.

**Supported request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_fields</td>
<td>A comma-separated list of fields to return in the response</td>
</tr>
<tr>
<td>sysparm_relation_limit</td>
<td>The maximum number of relations to return. Default is 1000.</td>
</tr>
<tr>
<td>sysparm_relation_offset</td>
<td>A number of records to exclude from the relations query. Default is 0.</td>
</tr>
</tbody>
</table>

**Headers**

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see *Supported REST API headers*.

**Request headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Response headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Status codes**

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see *REST response codes*.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
Response body

The API returns these JSON or XML elements in the response body.

```json
{
"result": {
"attributes": {
"cpu_manufacturer": {
"link": "http://localhost:8080/api/now/table/core_company/7aad6d00c611228400f00e0f80b67d2d",
"value": "7aad6d00c611228400f00e0f80b67d2d"
},
"sys_updated_on": "2017-06-28 07:52:53",
"ram": "2048",
"skip_sync": "false",
"sys_updated_by": "system",
"sys_created_on": "2006-07-09 13:59:06",
"sys_domain": {
"link": "http://localhost:8080/api/now/table/sys_user_group/global",
"value": "global"}
},
"outbound_relations": [{
"sys_id": "e7bac2d10a0a0aa700ba523f2a4ccf44",
"type": {
"link": "http://localhost:8080/api/now/table/cmdb_rel_type/e76b8c7b0a0a70082c9f7c2f9dc64",
"value": "e76b8c7b0a0a70082c9f7c2f9dc64"},
"target": {
"link": "http://localhost:8080/api/now/cmdb/instance/cmdb_ci/e7a0a9be0a0a7004b5fbd1d177144",
"value": "e7a0a9be0a0a7004b5fbd1d177144"}}],
"inbound_relations": [{
"sys_id": "5399a396c0a8014001b264c2b2e5",
"type": {
"link": "http://localhost:8080/api/now/table/cmdb_rel_type/1a9cb166f1571100a92eb60da2bce5c5",
"value": "1a9cb166f1571100a92eb60da2bce5c5"},
"target": {
"link": "http://localhost:8080/api/now/cmdb/instance/cmdb_ci/451047c6c0a8014000e0a6d9b9d76",
"value": "451047c6c0a8014000e0a6d9b9d76"}}],
"type": {
"link": "http://localhost:8080/api/now/table/cmdb_rel_type/1a9cb166f1571100a92eb60da2bce5c5",
"value": "1a9cb166f1571100a92eb60da2bce5c5"},
"target": {
"link": "http://localhost:8080/api/now/cmdb/instance/cmdb_ci/60bc4e22e08010e0124fb16964b989f",
"value": "60bc4e22e08010e0124fb16964b989f"}
}
}```
CMDB instance API - PATCH - /now/cmdb/instance/{classname}/{sys_id}

Replace the specified record with the message body. This API does not create/update/delete the associated relations.

URL format

Versioned URL: api/now/v1/cmdb/instance/{className}/{sys_id}
Default URL: api/now/cmdb/instance/{className}/{sys_id}

The path parameters.
- className - The CMDB class name.
- sys_id - The record sys_id.

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_fields</td>
<td>A comma-separated list of fields to return in the response</td>
</tr>
<tr>
<td>sysparm_relation_limit</td>
<td>The maximum number of relations to return. Default is 1000.</td>
</tr>
<tr>
<td>sysparm_relation_offset</td>
<td>A number of records to exclude from the relations query. Default is 0.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
**Status codes**

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see _REST response codes_.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Request body**

The API accepts these JSON or XML elements in the request body.

```
{"attributes": {
    "field1": "value1",
    "field2": "value2"},
source: "ServiceNow" }
```

**Response body**

The API returns these JSON or XML elements in the response body.

```
{"result": [{
    "sys_id": "3a290cc60a0a0bb400000bdb386af1cf",
    "name": "PS LinuxApp01"},
    {
    "sys_id": "3a5dd3dbc0a8ce0100655f1ec66ed42c",
    "name": "PS LinuxApp02"}]
```

**CMDB instance API - POST - /now/cmdb/instance/{classname}**

Create a single CI with associated outbound and inbound relations. If the class has a dependent identification rule defined, then the input payload must contain the sys_id of the parent CI.

**URL format**

Versioned URL: api/now/v1/cmdb/instance/{className}

Default URL: api/now/v1/cmdb/instance/{className}

The path parameter.

- className - The CMDB class name.
Supported request parameters

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_fields</td>
<td>A comma-separated list of fields to return in the response</td>
</tr>
<tr>
<td>sysparm_relation_limit</td>
<td>The maximum number of relations to return. Default is 1000.</td>
</tr>
<tr>
<td>sysparm_relation_offset</td>
<td>A number of records to exclude from the relations query. Default is 0.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Request body

The API accepts these JSON or XML elements in the request body.

```json
{attributes:
}
```
Response body

The API returns these JSON or XML elements in the response body.

```
{
  "result": [
    {
      "sys_id": "3a290cc60a0a0bb400000bdb386af1cf",
      "name": "PS LinuxApp01"
    },
    {
      "sys_id": "3a5dd3dbc0a8ce0100655f1ec66ed42c",
      "name": "PS LinuxApp02"
    }
  ]
}
```

CMDB instance API - POST - /now/cmdb/instance/{classname}/{sys_id}/relation

Add a relation to the specified CI. The specified CI is the parent.

URL format

Versioned URL: api/now/v1/cmdb/instance/{className}/{sys_id}/relation

Default URL: api/now/cmdb/instance/{className}/{sys_id}/relation

The path parameters.
- className - The CMDB class name.
- sys_id - The record sys_id.

Supported request parameters

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_fields</td>
<td>A comma-separated list of fields to return in the response</td>
</tr>
<tr>
<td>sysparm_relation_limit</td>
<td>The maximum number of relations to return. Default is 1000.</td>
</tr>
<tr>
<td>sysparm_relation_offset</td>
<td>A number of records to exclude from the relations query. Default is 0.</td>
</tr>
</tbody>
</table>
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see [Supported REST API headers](#).

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>The location of the created resource.</td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see [REST response codes](#).

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Request body

The API accepts these JSON or XML elements in the request body.

```json
{
  "outbound_relations": [
    {
      "target": "TARGET_SYS_ID",
      "type": "REL_TYPE_SYS_ID"
    }
  ],
  "inbound_relations": [
    {
      "target": "TARGET_SYS_ID",
      "type": "REL_TYPE_SYS_ID"
    }
  ]
}
```

**CMDB instance API - PUT - /now/cmdb/instance/{classname}/{sys_id}**

Replace the specified record with the message body. This API does not create/update/delete the associated relations.
URL format

Versioned URL: api/now/v1/cmdb/instance/{className}/{sys_id}
Default URL: api/now/cmdb/instance/{className}/{sys_id}

The path parameters.
- className - The CMDB class name.
- sys_id - The record sys_id.

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_fields</td>
<td>A comma-separated list of fields to return in the response</td>
</tr>
<tr>
<td>sysparm_relation_limit</td>
<td>The maximum number of relations to return. Default is 1000.</td>
</tr>
<tr>
<td>sysparm_relation_offset</td>
<td>A number of records to exclude from the relations query. Default is 0.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.
Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Request body

The API accepts these JSON or XML elements in the request body.

```json
{"attributes": {
    "field1": "value1",
    "field2": "value2"},
source: "ServiceNow"
}
```

Response body

The API returns these JSON or XML elements in the response body.

```json
{"result": [{
    "sys_id": "3a290cc60a0a0bb400000bdb386af1cf",
    "name": "PS LinuxApp01"},
    {
    "sys_id": "3a5dd3db0a8ce0100655f1ec66ed42c",
    "name": "PS LinuxApp02"
}]
}
```

CMDB meta API

Use the CMDB meta API to obtain meta data on a CMDB class.

These APIs require that the user have the ITIL role.

**CMDB meta API - GET - /now/cmdb/meta/{classname}**

Read meta data for a CMDB class.

**URL format**

Versioned URL: api/now/v1/cmdb/meta/{className}

Default URL: api/now/cmdb/meta/{className}

The path parameter.

- className - The CMDB class name.
Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see [Supported REST API headers](#).

**Request headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Response headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see [REST response codes](#).

**Status codes**

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response body

The API returns these JSON or XML elements in the response body.

```json
{"result": {
    "parent": "cmdb_ci",
    "children": [
        "cmdb_ci_appl_ora_oacore",
        "cmdb_ci_appl_ca",
        "cmdb_ci_db_mysql_clusternode"],
    "attributes": [{
        "label": "TCP port(s)",
        "type": "string"},
```
Email API

With the Email API you can receive and send email messages using REST.

Security

Users must have the email_api_send role to send email.

Email API - GET /now/email/{sys_id}

This method returns the record details of the specified email record.

URL format

Versioned URL: /api/now/v1/email/{sys_id}
Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully. If a valid query returned no results, the response body contains only an empty result array.</td>
</tr>
<tr>
<td>403</td>
<td>Indicates the record is not found or the requesting user does not have access to the record. Verify the user has the proper role and access permissions.</td>
</tr>
</tbody>
</table>

Request body

The API accepts these JSON or XML elements in the request body.
### Elements accepted in the request body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

### Response body

The API returns these JSON or XML elements in the response body.

### Elements returned in the response body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headers</td>
<td>Array consisting of the email message headers.</td>
</tr>
<tr>
<td>cc</td>
<td>Array consisting of the copied recipients of the email message. Maps to the copied field.</td>
</tr>
<tr>
<td>type</td>
<td>The current state of the email message as incoming or outgoing mail. Values include: received, received-ignored, send-failed, send-ignored, send-ready, sent</td>
</tr>
<tr>
<td>html</td>
<td>The HTML-enabled body of the email message. Maps to the body field.</td>
</tr>
<tr>
<td>bcc</td>
<td>Array consisting of the blind copied recipients of the email message. Maps to the blind_copied field.</td>
</tr>
<tr>
<td>subject</td>
<td>Specifies the subject of the email message. Maps to the subject field.</td>
</tr>
<tr>
<td>to</td>
<td>Array consisting of the direct recipients of the email message. Maps to the recipients field.</td>
</tr>
<tr>
<td>state</td>
<td>The processing state of the email message. Indicates whether system scheduled jobs have processed the email message. Values include: error, ignored, processed, ready</td>
</tr>
<tr>
<td>id</td>
<td>The sys_id of the Email record.</td>
</tr>
<tr>
<td>importance</td>
<td>The importance of the email message. Maps to the importance field.</td>
</tr>
</tbody>
</table>
Sample cURL request

curl "http://instance.service-now.com/api/now/email/06e095427f0022007f005212bdfa91b3" \
--request GET \
--header "Accept:application/json" \
--user 'user-name':'password'

```json
{
    "result": {
        "headers": {
            "X-ServiceNow-SysEmail-Version": "2",
            "X-ServiceNow-Source": "Notification-24e34b54c61122aa0108c1b7a33697cf"
        },
        "cc": [
            
        ],
        "subject": "Your incident INC00000005 has been closed",
        "state": "ready",
        "id": "06e095427f0022007f005212bdfa91b3",
        "importance": "",
        "text": "",
        "type": "send-ready",
        "html": "<html><head></head><body><div><p><font size="5" color="#808080" face="helvetica"><strong>Incident has been closed.</strong></font></p></div><div><p><font size="4" color="#808080" face="helvetica"><strong>Summary details</strong></font></p><p><font size="3" color="#808080" face="helvetica">Closed by: System Administrator</font></p><p><font size="3" color="#808080" face="helvetica">Closed notes: Fixed</font></p></div><div><p><font size="3" color="#808080" face="helvetica">You can view all the details of the incident by following the link below:<br /></p><a href="incident.do?sys_id=e8e875b0c0a80164009dc852b4d677d5&amp;sysparm_stack=incident_list.do?sysparm_query=active=true" style="background-color: #278efc;border: 1px solid #0368d4;color: #ffffff;font-size: 16px;font-family: Helvetica, Arial, sans-serif;text-decoration: none;border-radius: 3px;">Take me to the Incident</a><br /><br />
Thank you.</div></body></html>
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'http://instance.service-now.com/api/now/email/06e095427f0022007f005212bdaf91b3'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type': "application/json", 'Accept': "application/json"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)
```

```json
{
    "result": {
        "headers": {
            "X-ServiceNow-SysEmail-Version": "2",
            "X-ServiceNow-Source": "Notification-24e34b54c61122aa0108c1b7a33697cf"
        },
        "cc": [],
        "type": "send-ready",
        "html": "<html><head></head><body><div><p><font size="5" color="#808080" face="helvetica"><strong>Incident has been closed.</strong></font></p></div><div><p><font size="4" color="#808080" face="helvetica"><strong>Summary details</strong></font></p><p><font size="3" color="#808080" face="helvetica">Closed by: System Administrator</font></p><p><font size="3" color="#808080" face="helvetica">Closed notes: Fixed</font></p></div><div><p><font size="3" color="#808080" face="helvetica">You can view all the details of the incident by following the link below:</font></p><a href="incident.do?sys_id=e8e875b0c0a80164009dc852b4d677d5&sysparm_stack=incident_list.do?sysparm_query=active=true" style="background-color: #278efc;border: 1px solid #0368d4;color: #ffffff;font-size: 16px;font-family: Helvetica, Arial, sans-serif;text-decoration: none;border-radius: 3px;-webkit-border-radius: 3px;-moz-border-radius: 3px;display: inline-block;padding: 5px;">Take me to the Incident</a></div><br /><p><font size="3" color="#808080" face="helvetica">Thank you.</font></p>
```
Email API - POST /now/email

This method creates the email record specified in the request body.

URL format

Versioned URL: /api/now/v1/email

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

### Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request completed successfully.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates that there is an error with the request message. The response body contains information about the error.</td>
</tr>
<tr>
<td>403</td>
<td>Indicates that the requesting user does not have access to the record. Verify that the user has the proper role and access permissions.</td>
</tr>
<tr>
<td>500</td>
<td>Indicates that there is an unexpected or internal error.</td>
</tr>
</tbody>
</table>

Request body

The API accepts these JSON or XML elements in the request body to specify the content of the email message you want to create in the Email (sys_email) table.

### Elements accepted in the request body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
</table>
| to      | Specifies the direct recipients of the email message. Maps to the recipients field.  
(Required)  |

**Note:** You can only specify up to 100 addresses in this field.

| cc       | Specifies the copied recipients of the email message. Maps to the copied field. |

**Note:** You can only specify up to 100 addresses in this field.

| bcc      | Specifies the blind copied recipients of the email message. Maps to the blind_copied field. |

**Note:** You can only specify up to 100 addresses in this field.

<p>| subject  | Specifies the subject of the email message. Maps to the subject field. |</p>
<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Specifies the text-only body of the email message. Maps to the body_text field.</td>
</tr>
<tr>
<td>html</td>
<td>Specifies the HTML-enabled body of the email message. Maps to the body field.</td>
</tr>
<tr>
<td>importance</td>
<td>Specifies the importance of the email message. Maps to the importance field.</td>
</tr>
<tr>
<td>table_name</td>
<td>Specifies the table of the related record to which the email applies. Use this parameter to associate an email message to a particular related record elsewhere in the system.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This parameter also requires specifying the <code>table_record_id</code> parameter.</td>
</tr>
<tr>
<td>table_record_id</td>
<td>Specifies the target-related record to which the email applies. Use this parameter to associate an email message to a particular related record elsewhere in the system.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This parameter also requires specifying the <code>table_name</code> parameter.</td>
</tr>
<tr>
<td>headers</td>
<td>Specify an array of custom headers you want to include in the email message. Custom headers must begin with an <code>X-</code> prefix. These custom headers are reserved for ServiceNow only and produce a 400 status code error:</td>
</tr>
<tr>
<td></td>
<td>· X-ServiceNow</td>
</tr>
<tr>
<td></td>
<td>· X-Service-Now</td>
</tr>
</tbody>
</table>

**Response body**

The API returns these JSON or XML elements in the response body to describe the email record created.

**Response body elements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The sys_id of the Email record.</td>
</tr>
<tr>
<td>links</td>
<td>Array consisting of links to the Email record.</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| rel     | The type of link listed in the href parameter.  
  - **self**: The Email API GET request for the Email record.  
  - **status**: The Email API GET request for the Email record showing only the id, type, state, and error fields. |
| href    | Specifies the link to the Email record as an Email API GET request. |

**Sample cURL request**

```bash
curl "http://instance.service-now.com/api/now/email" \
--request POST \
--header "Accept:application/json" \
--header "Content-Type:application/json" \
--data "{
   \"to\": [
      \"User1 <user1@example.com>\", \
      \"User2 <user2@example.com>\"
   ],
   \"cc\": [
      \"User3 <user3@example.com>\", \
      \"User4 <user4@example.com>\"
   ],
   \"bcc\": [
      \"User5 <user5@example.com>\", \
      \"User6 <user6@example.com>\"
   ],
   \"subject\": \"Hello There\", 
   \"text\": \"Test Message\", 
   \"html\": \"<b>Test Message</b>\", 
   \"table_name\": \"incident\", 
   \"table_record_id\": "136b2140bd0312004d7d1371f1abbd6b",
   \"headers\": {
      \"X-Custom\": \"header\"
   }
}" \
--user 'admin':'admin'
```

```json
{
 "result": {
   "id": "b963219a44b02200964f63773cd6adfc",
   "links": [
   {
      "rel": "self",
      "href": "/now/v1/email/b963219a44b02200964f63773cd6adfc"
   },
   {
      "rel": "status",
      "href": "/now/v1/email/b963219a44b02200964f63773cd6adfc?sysparm_fields=id,type,state,error"
   }
 ]
}
```
Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'http://localhost:8080/api/now/v1/email'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/json","Accept":"application/json"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd), headers=headers ,data="{
  "to": [
    "User1 <user1@example.com>",
    "User2 <user2@example.com>"
  ],
  "cc": [
    "User3 <user3@example.com>",
    "User4 <user4@example.com>"
  ],
  "bcc": [
    "User5 <user5@example.com>",
    "User6 <user6@example.com>"
  ],
  "subject": "Hello There",
  "text": "Test Message",
  "html": "<b>Test Message</b>",
  "table_name": "incident",
  "table_record_id": "136b2140bd0312004d7d1371f1abbdb6",
  "headers": {"X-Custom": "header"}
}
"")

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

{
  "result": {
    "id": "b963219a44b02200964f63773cd6adfc",
    "links": [
      {"rel": "self",
      "href": "/now/v1/email/b963219a44b02200964f63773cd6adfc"
    ]
  }
}
```
IdentifyReconcile API

The IdentifyReconcile API uses the Identification and Reconciliation framework to minimize creation of duplicate CIs and to reconcile CI attributes by only accepting information from authorized sources when updating the CMDB.

For Kingston Patch 4 and subsequent releases, users must have the itil or asset role to use this API.

IdentifyReconcile API - POST /now/identifyreconcile

Insert or update configuration items in the CMDB based on identification and reconciliation rules. Use this API instead of updating the CMDB directly.

URL format

Versioned URL: /api/now/v1/identifyreconcile

Default URL: /api/now/identifyreconcile

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_data_source</td>
<td>Identifies the source of the CI information. This must be one of the choice values defined for the discovery_source field of the cmdb_ci table.</td>
</tr>
</tbody>
</table>

Headers

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Specify application/json</td>
</tr>
<tr>
<td>Content-Type</td>
<td>Specify application/json</td>
</tr>
</tbody>
</table>
Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
</table>

Input payload

A JSON formatted string of configuration items to be added or updated. Each input string is in the format `items: ([{}], relations: ({}))`, where each item within the items and relations lists contains name-value pairs.

The possible name-value pairs within the items list are:

- `className` - the sys_class_name of the CI to be created or updated.
- `values: {}` - the field information for the CI as name-value pairs, where the name is the field name. When updating reference fields, the value must be the referenced sys_id.
- `lookup: ({}[])` - a list of records with each item having name-value pairs like the items list.

The possible name-value pairs within the relations list are:

- `parent` - index of the parent item in the dependency relation
- `child` - index of the child item in the dependency relation
- `type` - the relationship type. This is one of the name field values from the cmdb_rel_type table.

Sample input payload

The sample input is a JSON string that contains a list of configuration items and a list of relationships that exist between these configuration items.

```json
{"items": [ {
"className": "cmdb_ci_web_server",
"values": {
"name": "apache linux den 200",
"running_process_command": "xyz",
"running_process_parameters": "abc",
"tcp_port": "3452"
},
"tcp_port": "3452"
},
{"className": "cmdb_ci_linux_server",
"values": {
"name": "linux100", 
"ram": "2048"
}}],
"relations": [{
"parent": "0", 
"child": "1", 
"type": "Runs on::Runs"
}]}
```

Response payload

A JSON formatted string that is a list of results for the configuration items in the input string. Each result string is in the format `items: ([{}], relations: ({}))`, where each item within the items and relations lists contains name-value pairs.

The possible name-value pairs within the items list are:

- `className` - the sys_class_name for the CI to be updated or created.
- `operation`, which is one of INSERT, UPDATE, UPDATE_WITH_UPGRADE, UPDATE_WITH_DOWNGRADE, UPDATE_WITH_SWITCH, DELETE, NO_CHANGE
- `sysId` - the sys_id of the CI that was updated or created.
- `relatedSysIds` - a list of sys_id values of CIs used during lookup based identification.
- `maskedAttributes` - a list of attributes whose update by a non-authoritative data source gets skipped as defined by the Reconciliation Rules.
- identifierEntrySysId - sys_id of identifier entry used during matching.
- errors - a list of errors in the format of (error, message string), where error can be ABANDONED, INVALID_INPUT_DATA, IDENTIFICATION_RULE_MISSING, IDENTIFICATION_RULE_FOR_LOOKUP_MISSING, NO_LOOKUP_RULES_FOR_DEPENDENT_CI, NO_CLASS_NAME_FOR_INDEPENDENT_CI, MISSING_DEPENDENCY, MULTIPLE_DEPENDENCIES, MULTIPLE_DUPLICATE_RECORDS, RELATION_CHAIN_ENDS_AT_QUALIFIER, QUALIFICATION_LOOP, TYPE_CONFLICT_IN_QUALIFICATION, MULTI_MATCH, REQUIRED_ATTRIBUTE_EMPTY, RECLASSIFICATION_NOT_ALLOWED
- duplicateIndices - a list of indexes of items that are duplicates of the current item.
- identificationAttempts - a list of attempts in the format of (attributes, identifierName, attemptResult, searchOnTable) where
  - attributes - the attributes used during identification
  - identifierName - the CI identifier to which this identifier belongs
  - attemptResult - one of SKIPPED, NO_MATCH, MATCHED, MULTI_MATCH
  - searchOnTable - the table searched during the identification process.

The possible name-value pairs within the relations list are:
- className - the relationship CI's class name and is always cmdb_rel_ci
- operation - one of INSERT, UPDATE, NO_CHANGE
- sysId - the sys_id of the relationship CI inserted or updated

Sample responses

Line feeds have been added for presentation.

```json
{ "items": [ { "className": "cmdb_ci_web_server", "operation": "UPDATE", "sysId": "5f8af237c0a8010e01a932999468b83a", "identifierEntrySysId": "8985a23ec3f00200d8d4bea192d3ae08", "identificationAttempts": [ { "attributes": [ "running_process_command", "running_process_parameters" ], "identifierName": "Application Rule", "attemptResult": "MATCHED", "searchOnTable": "cmdb_ci_appl" } ] }, { "className": "cmdb_ci_linux_server", "operation": "UPDATE", "sysId": "53958ff0c0a801640171ec76aa0c8f86", "identifierEntrySysId": "556eb250c3400200d8d4bea192d3ae92", "identificationAttempts": [ { "attributes": [ "serial_number" ], "identifierName": "Hardware Rule", "attemptResult": "SKIPPED", "searchOnTable": "cmdb_ci_hardware" } ] } ] }
```
Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

IdentifyReconcile API - POST /now/identifyreconcile/query

Determines the operation (insert/update) that will be performed with the specified payload without committing the operation in the database.

URL format

Versioned URL: /api/now/v1/identifyreconcile/query
Default URL: /api/now/identifyreconcile/query

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Specify application/json</td>
</tr>
<tr>
<td>Content-type</td>
<td>Specify application/json</td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
</tr>
</tbody>
</table>

Request body

The API accepts a JSON formatted string of configuration items to be added or updated. Each input string is in the format ‘items: ({}), relations:({})’, where each item within the items and relations lists contains name-value pairs.

Elements accepted in the request body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
</table>
| items: ({}), relations:({}) | The possible name-value pairs within the items list are:  
  - className - the sys_class_name of the CI to be created or updated.  
  - values:{} - the field information for the CI as name-value pairs, where the name is the field name. When updating reference fields, the value must be the referenced sys_id.  
  - lookup:({}) - a list of records with each item having name-value pairs like the items list.  
| relations:({}) | The possible name-value pairs within the relations list are:  
  - parent - index of the parent item in the dependency relation  
  - child - index of the child item in the dependency relation  
  - type - the relationship type. This is one of the name field values from the cmdb_rel_type table. |

Response body

The API returns a JSON formatted string that is a list of results. Each result string is in the format ‘items: ({}), relations:({})’, where each item within the items and relations lists contains name-value pairs.
## Elements returned in the response body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>items: [{})</td>
<td>The possible name-value pairs within the items list are:</td>
</tr>
<tr>
<td>- className - the sys_class_name for the CI that was updated or created.</td>
<td></td>
</tr>
<tr>
<td>- operation, which is one of INSERT, UPDATE, UPDATE_WITH_UPGRADE, UPDATE_WITH_DOWNGRADE, UPDATE_WITH_SWITCH, DELETE, NO_CHANGE</td>
<td></td>
</tr>
<tr>
<td>- sysId - the sys_id of the CI that was updated or created.</td>
<td></td>
</tr>
<tr>
<td>- relatedSysIds - a list of sys_id values of CIs used during lookup based identification.</td>
<td></td>
</tr>
<tr>
<td>- identifierEntrySysId - sys_id of identifier entry used during matching.</td>
<td></td>
</tr>
<tr>
<td>- errors - a list of errors in the format of (error, message string), where error can be ABANDONED, INVALID_INPUT_DATA, IDENTIFICATION_RULE_MISSING, IDENTIFICATION_RULE_FOR_LOOKUP_MISSING, NO_LOOKUP_RULES_FOR_DEPENDENT_CI, NO_CLASS_NAME_FOR_INDEPENDENT_CI, MISSING_DEPENDENCY, MULTIPLE_DEPENDENCIES, MULTIPLE_DUPLICATE RECORDS, RELATION_CHAIN_ENDS_AT_QUALIFIER, QUALIFICATION_LOOP, TYPE_CONFLICT_IN_QUALIFICATION, MULTI_MATCH, REQUIRED_ATTRIBUTE EMPTY, RECLASSIFICATION_NOT_ALLOWED</td>
<td></td>
</tr>
<tr>
<td>- duplicateIndices - a list of indexes of items that are duplicates of the current item.</td>
<td></td>
</tr>
<tr>
<td>- identificationAttempts - a list of attempts in the format of (attributes, identifierName, attemptResult, searchOnTable) where</td>
<td></td>
</tr>
<tr>
<td>- attributes - the attributes of identifier entry used during identification</td>
<td></td>
</tr>
<tr>
<td>- identifierName - the CI identifier to which this identifier entry belongs</td>
<td></td>
</tr>
<tr>
<td>- attemptResult - one of SKIPPED, NO_MATCH, MATCHED, MULTI_MATCH</td>
<td></td>
</tr>
<tr>
<td>- searchOnTable - the table searched during the identification process.</td>
<td></td>
</tr>
</tbody>
</table>
Import Set API

The Import Set API allows you to interact with import set tables.

Overview

The API transforms incoming data based on associated transform maps. The import set API supports synchronous transforms. The Import Set API mirrors the existing SOAP interface.

Security

Access to tables via the REST API is restricted by BasicAuth. To allow access to tables without any authentication or authorization, add the table name to sys_public.list. ACLs defined on tables are still enforced, and it is the administrator's responsibility to deactivate ACLs.

Import Set API - GET /now/import/{tableName}/{sys_id}

This method retrieves the specified import staging record and resulting transformation result.

URL format

Versioned URL: /api/now/v1/import/{tableName}/{sys_id}
Default URL: /api/now/import/{tableName}/{sys_id}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

The possible name-value pairs within the relations list are:
- className - the relationship CI's class name and is always cmdb_rel_ci
- operation - one of INSERT, UPDATE, NO_CHANGE
- sysId - the sys_id of the relationship CI inserted or updated
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

<table>
<thead>
<tr>
<th>Request headers</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>Description</td>
</tr>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response headers</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>Description</td>
</tr>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status codes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Status code</td>
<td>Description</td>
</tr>
<tr>
<td>200</td>
<td>Indicates the query ran successfully.</td>
</tr>
<tr>
<td>404</td>
<td>Indicates the specified resource was not available. As import set tables are deleted frequently based on a schedule, GET requests may return 404 NotFound responses if the transformation result no longer exists.</td>
</tr>
</tbody>
</table>

Sample cURL request

curl "https://instance.service-now.com/api/now/import/imp_user/e2928be64f411200adf9f8e18110c777" \
--request GET \
--header "Accept:application/json" \
--user 'admin':'admin' \\

```json
{

"import_set": "ISET0010001",
"staging_table": "imp_user",
"result": [

{
"transform_map": "User",
"table": "sys_user",
"display_name": "name",
"display_value": "John Public",
"record_link": "https://instance.service-now.com/api/now/table/sys_user/ea928be64f411200adf9f8e18110c777",

```
Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/import/imp_user/e2928be64f411200adf9f8e18110c777'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/xml","Accept":"application/xml"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)
```

```xml
<response>
  <import_set>ISET0010001</import_set>
  <staging_table>imp_user</staging_table>
  <result>
    <display_name>name</display_name>
    <display_value>John Public</display_value>
    <status>inserted</status>
    <sys_id>ea928be64f411200adf9f8e18110c777</sys_id>
    <record_link>https://instance.service-now.com/api/now/table/sys_user/ea928be64f411200adf9f8e18110c777</record_link>
  </result>
</response>
```

**Import Set API - POST /now/import/(tableName)**

This method inserts incoming data into a specified staging table and triggers transformation based on predefined transform maps in the import set table.
URL format

Versioned URL: /api/now/v1/import/<staging table_name>
Default URL: /api/now/import/<staging table_name>

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>The URL of the created resource.</td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Indicates the import completed successfully.</td>
</tr>
</tbody>
</table>

Transforming inserted records

Transformation occurs synchronously. For each transform map that you define, the responses include transformation results such as information on the target records.
Note: The status_message and error_message fields on transformation scripts are processed and returned in response, along with any custom response fields.

Sample cURL request

curl "https://instance.service-now.com/api/now/import/imp_user" \
--request POST \
--header "Accept:application/json" \
--header "Content-Type:application/json" \
--data 
"{"first_name':'John','last_name':'Public','user_id':'john.public','email':'john.public@company.com'" \
--user 'admin':'admin'

{
   "import_set": "ISET0010001",
   "staging_table": "imp_user",
   "result": [
   {
      "transform_map": "User",
      "table": "sys_user",
      "display_name": "name",
      "display_value": "John Public",
      "record_link": "https://instance.service-now.com/api/now/table/sys_user/ea928be64f411200adf9f8e18110c777",
      "status": "inserted",
      "sys_id": "ea928be64f411200adf9f8e18110c777"
   }
   ]
}

Sample Python request

#Need to install requests package for python
#easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/import/imp_user'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/xml","Accept":"application/xml"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd),
    headers=headers, data="<request><entry><first_name>John</first_name><last_name>Public</last_name><user_id>john.public</user_id><email>john.public@company.com</email></entry></request>"

# Check for HTTP codes other than 200
if response.status_code != 200:
Performance Analytics API

The Performance Analytics REST API enables you to query data about Performance Analytics scorecards.

The Performance Analytics API supports only the GET action. Performance Analytics queries never update records.

**Performance Analytics API - GET /now/pa/scorecards**

This method retrieves Performance Analytics scorecard details.

**URL format**

- **Versioned URL**: /api/now/v1/pa/scorecards
- **Default URL**: /api/now/pa/scorecards

```python
print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)
```
## Supported parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_uuid</td>
<td>Enter a colon-separated list of sys_id values to specify which indicators, breakdowns, and aggregates to query. The parameter follows this format: &lt;indicator sys_id&gt;:&lt;breakdown sys_id&gt;:&lt;element sys_id&gt;:&lt;aggregate sys_id&gt; The parameter must begin with the sys_id of an indicator record. Optionally, you can append the sys_id values of a breakdown and breakdown element to group the response based on the breakdown, and the sys_id of an aggregate to apply that aggregate. You can use a breakdown with an aggregate, or use only one.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If an indicator is configured to use a Default time series, all scorecards for that indicator use the selected aggregate. See Performance Analytics API Examples for examples of fully-constructed sysparm_uuid values.</td>
</tr>
<tr>
<td>sysparm_breakdown</td>
<td>Enter the sys_id of a breakdown to return chart information organized as defined by the breakdown. For example, enter the sys_id of a priority breakdown to return separate task chart information for each priority value, such as Number of open incidents / Priority / 2 - High.</td>
</tr>
<tr>
<td>sysparm_include_scores</td>
<td>Set to <code>true</code> to return all scores for a scorecard. If a value is not specified, this parameter defaults to false and returns only the most recent score value.</td>
</tr>
<tr>
<td>sysparm_include_aggregates</td>
<td>Set to <code>true</code> to always return all available aggregates for an indicator, including when an aggregate has already been applied. If a value is not specified, this parameter defaults to false and returns no aggregates.</td>
</tr>
<tr>
<td>sysparm_include_available_breakdowns</td>
<td>Set to <code>true</code> to return all available breakdowns for an indicator. If a value is not specified, this parameter defaults to false and returns no breakdowns.</td>
</tr>
<tr>
<td>sysparm_include_available_aggregates</td>
<td>Set to <code>true</code> to return all available aggregates for an indicator when no aggregate has been applied. If a value is not specified, this parameter defaults to false and returns no aggregates.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_display_value</td>
<td>Data retrieval operation for reference and choice fields. Based on this value, retrieves the display value and/or the actual value from the database. Valid values: • true: returns display values for all fields. • false: returns actual values from the database. • all: returns both actual and display values.</td>
</tr>
<tr>
<td></td>
<td>Default: false</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> There is no preferred method for setting this parameter. However, specifying the display value may cause performance issues since it is not reading directly from the database and may include referencing other fields and records. For more information on display values and actual values, see Table API FAQs (KB0534905).</td>
</tr>
<tr>
<td>sysparm_exclude_reference_link</td>
<td>Set to true to hide additional information provided for reference fields, such as the URI to the reference resource.</td>
</tr>
<tr>
<td>sysparm_favorites</td>
<td>Set to true to return only scorecards that are favorites of the querying user.</td>
</tr>
<tr>
<td>sysparm_key</td>
<td>Set to true to return only scorecards for key indicators.</td>
</tr>
<tr>
<td>sysparm_target</td>
<td>Set to true to return only scorecards that have a target.</td>
</tr>
<tr>
<td>sysparm_display</td>
<td>Set to true to return only scorecards where the indicator <em>Display</em> field is selected. Set this parameter to <em>all</em> to return scorecards with any Display field value. This parameter is true by default.</td>
</tr>
<tr>
<td>sysparm_contains</td>
<td>Enter a comma-separated list of names or descriptions to return only scorecards with a matching value.</td>
</tr>
<tr>
<td>sysparm_tags</td>
<td>Enter a comma-separated list of sys_id values to return only scorecards with a matching sys_id.</td>
</tr>
<tr>
<td>sysparm_per_page</td>
<td>Enter the maximum number of scorecards each query can return. By default this value is 10, and the maximum is 100.</td>
</tr>
<tr>
<td>sysparm_page</td>
<td>Specify the page number. For example, when querying 20 scorecards with the default sysparm_per_page value (10), specify a sysparm_page value of 2 to retrieve scorecards 11-20.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sysparm_sortby</td>
<td>Specify the value to use when sorting results. Valid values for this parameter are value, change, changeperc, gap, gapperc, duedate, name, order, default, group, indicator_group, frequency, target, date, trend, bullet, and direction. By default, queries sort records by value.</td>
</tr>
<tr>
<td>sysparm_sortdir</td>
<td>Specify the sort direction, ascending or descending. By default, queries sort records in descending order. Set this parameter to asc to sort in ascending order.</td>
</tr>
<tr>
<td>sysparm_from</td>
<td>Specify the earliest date to return scores from. Only scores from this date or later are returned. The date format must match the ISO-8601 standard.</td>
</tr>
<tr>
<td>sysparm_to</td>
<td>Specify the latest date to return scores from. Only scores from this date or earlier are returned. The date format must match the ISO-8601 standard.</td>
</tr>
<tr>
<td>sysparm_step</td>
<td>Specify a numeric value to skip scores, based on the indicator frequency. For example, specify a value of 3 to return only scores from every third day for a daily indicator, or from every third week for a weekly indicator.</td>
</tr>
<tr>
<td>sysparm_limit</td>
<td>Specify the maximum number of scores to return.</td>
</tr>
<tr>
<td>sysparm_elements_filter</td>
<td>Specify the sys_id of an elements filter to apply that filter to the returned data.</td>
</tr>
<tr>
<td>sysparm_breakdown_relation</td>
<td>Specify the sys_id of a breakdown relation to break down the returned data using that relation. You can view available breakdown relations by setting the sysparm_include_available_breakdowns parameter to true.</td>
</tr>
<tr>
<td>sysparm_include_score_notes</td>
<td>Set this parameter to true to return all notes associated with the score. The note element contains the note text as well as the author and timestamp when the note was added.</td>
</tr>
<tr>
<td>sysparm_include_realtime</td>
<td>Set this parameter to true to return the realtime_enabled element which indicates if real-time scores are enabled for the indicator, and the realtime_value element which contains the real-time score value. This parameter is not supported for formula indicators.</td>
</tr>
<tr>
<td>sysparm_include_target_color_scheme</td>
<td>Set this parameter to true to return the target_color_scheme element that contains the minimum and maximum values, and the color of each section of the target color scheme for the scorecard.</td>
</tr>
<tr>
<td>sysparm_include_forecast_scores</td>
<td>Set this parameter to true to return the forecast_scores element that contains an array of date-value pairs that define the forecast data for the scorecard.</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_include_trendline_scores</td>
<td>Set this parameter to <code>true</code> to return the <code>trendline_scores</code> element that contains an array of date-value pairs that define the scorecard trendline.</td>
</tr>
</tbody>
</table>

### Headers

**Request headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

### Status codes

**Status codes**

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the query ran successfully.</td>
</tr>
</tbody>
</table>

### Service Catalog API

The Service Catalog API lets you access Service Catalog configuration and actions from within the Service Portal.

**Service Catalog API - DELETE /sn_sc/servicecatalog/cart/{cart_item_id}**

This method deletes the items in the cart for a given sys_id.

**URL format**

**Versioned URL:** `/api/sn_sc/v1/servicecatalog/cart/{cart_item_id}`

**Default URL:** `/api/sn_sc/servicecatalog/cart/{cart_item_id}`

**Supported request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cart_item_id</td>
<td>Represents the sys_id of the item in the cart.</td>
</tr>
</tbody>
</table>
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>Indicates that the request completed successfully and the given item is deleted from the cart.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates that either the given cart_item_id is invalid or the user does not have access to the cart item.</td>
</tr>
</tbody>
</table>

Sample cURL request

```bash
curl "https://instance.service-now.com/api/sn_sc/v1/servicecatalog/cart/1f4cc597db2112006430150f0b8f5fc" \
--request DELETE \
--header "Accept:application/json" \
--user 'admin':'admin'
```

Sample Python request

```python
#Need to install requests package for python
#easy_install requests
import requests

# Set the request parameters
```
url = 'https://instance.service-now.com/api/sn_sc/v1/servicecatalog/cart/1f4cc597db21120064301150f0b8f5fc'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type':"application/json","Accept" : "application/json"}

# Do the HTTP request
response = requests.delete(url, auth=(user, pwd), headers=headers )

# Check for HTTP codes other than 204
if response.status_code != 204:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

None

Service Catalog API - DELETE /sn_sc/servicecatalog/cart/{sys_id}/empty

This method deletes the cart and contents of the cart for a given user role and sys_id.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/cart/{sys_id}/empty
Default URL: /api/sn_sc/servicecatalog/cart/{sys_id}/empty

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_id</td>
<td>Indicates the sys_id of the cart to empty.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.
Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>Indicates that the request completed successfully and the items in the cart are successfully checked out.</td>
</tr>
</tbody>
</table>
| 400          | Indicates an error for one of the following reasons:  
  - Invalid CartId.  
  - User does not have admin/catalog_admin role, and trying to empty another user cart. |

User roles

User roles to delete cart and cart contents

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin/catalog_admin</td>
<td>A user with this role can empty any user cart.</td>
</tr>
<tr>
<td>admin</td>
<td>A user with this role can delete the cart after all the contents of the cart are deleted.</td>
</tr>
<tr>
<td>catalog_admin</td>
<td>A user with this role can delete all the items in the cart.</td>
</tr>
</tbody>
</table>

Sample cURL request

```
curl "https://instance.service-now.com/api/sn_sc/v1/servicecatalog/cart/checkout" \  
--request DELETE \  
--header "Accept:application/json" 
```
Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/v1/servicecatalog/cart/checkout'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type': 'application/json', 'Accept': 'application/json'}

# Do the HTTP request
response = requests.delete(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 204
if response.status_code != 204:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)
```

None

Service Catalog API - GET /sn_sc/servicecatalog/cart

This method retrieves the default list of cart contents, cart details, and price shown on the two-step checkout page.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/cart
Default URL: /api/sn_sc/servicecatalog/cart
Supported request parameters

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request completed successfully. If there is no item in the cart, it returns cart basic information and pricing for user. If cart contains any items, then the information about the items and their individual pricing is also included.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates that the cart is empty and cannot check out.</td>
</tr>
</tbody>
</table>

Response body

Following are JSON or XML response values for two-step checkout page details.
### Response body values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cart_id</td>
<td>Specifies the sys_id of the cart.</td>
</tr>
<tr>
<td>subtotal_price</td>
<td>Specifies the subtotal of the cart.</td>
</tr>
<tr>
<td>subtotal_recurring_frequency</td>
<td>Specifies the recurring frequency subtotal of the cart.</td>
</tr>
<tr>
<td>subtotal_recurring_price</td>
<td>Specifies the recurring price subtotal of the cart.</td>
</tr>
<tr>
<td>total_title</td>
<td>Specifies the title for total field on page.</td>
</tr>
<tr>
<td>show_subtotal_price</td>
<td>Specifies Boolean value to show or not show subtotal price.</td>
</tr>
<tr>
<td>subtotal_title</td>
<td>Specifies the title for subtotal field on page.</td>
</tr>
<tr>
<td>monthly, weekly, daily</td>
<td>Array of item details classified by recurring frequencies of the items in the cart.</td>
</tr>
<tr>
<td></td>
<td>- subtotal_price: Specifies the subtotal of the frequency block.</td>
</tr>
<tr>
<td></td>
<td>- subtotal_recurring_frequency: Specifies the recurring frequency subtotal of the frequency block.</td>
</tr>
<tr>
<td></td>
<td>- subtotal_recurring_price: Specifies the recurring price subtotal of the frequency block.</td>
</tr>
<tr>
<td></td>
<td>- total_title: Title for total field on the frequency block.</td>
</tr>
<tr>
<td>items</td>
<td>Array consisting of details of all the items in the cart.</td>
</tr>
<tr>
<td></td>
<td>- catalog_item_id: sys_id of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- variables: Object consisting of all the variable names and corresponding values for the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- quantity: Quantity of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- localized_price: Price of the item in local currency.</td>
</tr>
<tr>
<td></td>
<td>- price: Price of the item.</td>
</tr>
<tr>
<td></td>
<td>- localized_recurring_price: Recurring price of the item in local currency.</td>
</tr>
<tr>
<td></td>
<td>- recurring_price: Recurring price of the item.</td>
</tr>
<tr>
<td></td>
<td>- recurring_frequency: Recurring frequency of the item.</td>
</tr>
<tr>
<td></td>
<td>- item_name: Name of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- cart_item_id: sys_id of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- delivery_time: Delivery time of the item.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Sample cURL request

curl "https://instance.service-now.com/api/sn_sc/v1/servicecatalog/cart" \
--request GET \
--header "Accept:application/json" \
--user 'admin':'admin'

Sample Python request

#Need to install requests package for python
#easy_install requests

import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/v1/servicecatalog/cart'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type':"application/xml","Accept":"application/xml"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:',response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()

<?xml version="1.0" encoding="UTF-8" ?>
<result>
  <cart_id>e7a44997db21120064300150f0b8f517</cart_id>
  <subtotal_price>$1,599.98</subtotal_price>
  <subtotal_recurring_frequency>Monthly</subtotal_recurring_frequency>
  <subtotal_recurring_price>$58.00</subtotal_recurring_price>
  <total_title>Total</total_title>
  <monthly>
    <subtotal_price>$1,599.98</subtotal_price>
    <subtotal_recurring_frequency>Monthly</subtotal_recurring_frequency>
    <subtotal_recurring_price>$58.00</subtotal_recurring_price>
  </monthly>
  <total_title>Total</total_title>
  <items>
    <catalog_item_id>0d08837237153000158bbfc8bcbe5d02</catalog_item_id>
  </items>
  <variables>
<Monthly data allowance>500MB</Monthly data allowance>
<Storage>64GB</Storage>
<Color>Black</Color>
<Allocated carrier>AT & T Mobility</Allocated carrier>
<br /></variables>
<quantity>2</quantity>
<localized_price>$799.99</localized_price>
<price>$799.99</price>
<recurring_frequency>Monthly</recurring_frequency>
<localized_recurring_price>$30.00</localized_recurring_price>
<recurring_price>$29.00</recurring_price>
</items>
<cart_item_id>1f4cc597db21120064301150f0b8f5fc</cart_item_id>
<delivery_time>2 Days</delivery_time>
</items>
<subtotal_title>Subtotal</subtotal_title>
</monthly>
<subtotal_title>Subtotal</subtotal_title>
</result>

Service Catalog API - GET /sn_sc/servicecatalog/cart/delivery_address/{user_id}

This method retrieves the shipping address of the requested user.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/cart/delivery_address/{user_id}
Default URL: /api/sn_sc/servicecatalog/cart/delivery_address/{user_id}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_id</td>
<td>Indicates sys_id of the user, whose shipping address is retrieved.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.
Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Specifies the request completed successfully and returns the shipping address of the user.</td>
</tr>
<tr>
<td>400</td>
<td>Specifies that the user id is invalid.</td>
</tr>
</tbody>
</table>

Sample cURL request

```bash
curl "https://instance.service-now.com/api/sn_sc/v1/servicecatalog/cart/delivery_address/62826bf03710200044e0bfc8bcbe5df1" 
--request GET 
--header "Accept:application/json" 
--user 'admin':'admin'

{   "result": "\nBrasilia, \nBrasil"
}
```

Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/v1/servicecatalog/cart/delivery_address/62826bf03710200044e0bfc8bcbe5df1'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'
```
Service Catalog API - GET /sn_sc/servicecatalog/catalogs

This method retrieves a list of catalogs to which the user has access.

**URL format**

**Versioned URL:** /api/sn_sc/v1/servicecatalog/catalogs

**Default URL:** /api/sn_sc/servicecatalog/catalogs

**Supported request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_limit</td>
<td>Specify the number of entities in the response. This value applies to categories and items.</td>
</tr>
<tr>
<td>sysparm_text</td>
<td>Specify query text for search support. (Required)</td>
</tr>
<tr>
<td>sysparm_view</td>
<td>Specify the device to display the item (desktop, mobile, or both).</td>
</tr>
</tbody>
</table>

**Headers**

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.
### Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

### Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

### Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see [REST response codes](https://service-now.com/docs/).

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request completed successfully. If a valid query returned no results, the response body contains only an empty result array.</td>
</tr>
</tbody>
</table>

### Sample cURL request

```bash
curl "https://instance.service-now.com/api/sn_sc/servicecatalog/catalogs?sysparm_limit=10" \
--request GET \
--header "Accept:application/json" \
--user 'admin':'admin'
```

```json
{
    "result": [ 
        {
            "title": "Technical Catalog",
            "sys_id": "742ce428d7211100f2d224837e61036d",
            "has_categories": true,
            "has_items": true,
            "description": "Products and services for the IT department",
            "desktop_image": "7a7c8271475211002ee987e8dee4906d.iix"
        },
        {
            "title": "Service Catalog",
            "sys_id": "e0d08b13c3330100c8b837659bba8fb4",
            "has_categories": true,
            "has_items": true,
            "description": "Service Catalog - IT Now",
            "desktop_image": "adbcc271475211002ee987e8dee49001.iix"
        }
    ]
}
```
Sample Python request

```python
# Need to install requests package for python
# easy_install requests

import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/v1/
    servicecatalog/catalogs?sysparm_limit=10'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type': 'application/json',
           'Accept': 'application/json'}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:',
          response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)
```

**Service Catalog API - GET /sn_sc/servicecatalog/catalogs/{sys_id}**

This method retrieves all the information about a requested catalog.

**URL format**

Versioned URL: /api/sn_sc/v1/servicecatalog/catalogs/{sys_id}
Default URL: /api/sn_sc/servicecatalog/catalogs/{sys_id}
Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_view</td>
<td>Defines the device that displays the Item:</td>
</tr>
<tr>
<td></td>
<td>· Desktop(desktop)</td>
</tr>
<tr>
<td></td>
<td>· Mobile(mobile)</td>
</tr>
<tr>
<td></td>
<td>· Desktop and Mobile(both)</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request has completed successfully and returns all the details of the catalog.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates that request is invalid for one of the following reasons:</td>
</tr>
<tr>
<td></td>
<td>· User does not have access to the catalog.</td>
</tr>
<tr>
<td></td>
<td>· Invalid sys_id.</td>
</tr>
</tbody>
</table>
### Response body values

Following are JSON or XML response parameters for a request completed successfully. The response consists of details of the catalog item.

**Response parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>Title of the catalog.</td>
</tr>
<tr>
<td>sys_id</td>
<td>SysId of the catalog.</td>
</tr>
<tr>
<td>has_categories</td>
<td>Returns true if the catalog has any categories created in it otherwise returns false.</td>
</tr>
</tbody>
</table>
| categories | A list of categories with the following details for each category:  
  - title: title of the category.  
  - description: description about the category.  
  - sys_id: sysId of the category.  
  - header_image: header image of the category. |
| has_items | Returns true if the catalog has any items associated with it otherwise returns false. |
| description | Description about the catalog. |
| desktop_image | The source of the image used to display with catalog. |

**Sample cURL request**

```bash
curl "https://instance.service-now.com/api/sn_sc/servicecatalog/catalogs/e0d08b13c3330100c8b837659bba8fb4" 
--request GET 
--header "Accept:application/json" 
--user 'admin':'admin'
```

**Sample Python request**

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/servicecatalog/catalogs/e0d08b13c3330100c8b837659bba8fb4'
```
# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type': 'application/xml','Accept': 'application/xml'}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

<?xml version="1.0" encoding="UTF-8"?>
<response>
    <result>
        <sys_id>e0d08b13c33301000c8b837659ba8fb4</sys_id>
        <desktop_image>adbcc271475211002ee987e8dee49001.iix</desktop_image>
        <has_categories />
        <description>Service Catalog - IT Now</description>
        <categories>
            <header_image />
            <sys_id>e15706fc0a0a0aa7007fc21e1ab70c2f</sys_id>
            <description>Your IT gateway. Report issues and submit requests.</description>
            <title>Can We Help You?</title>
        </categories>
        <categories>
            <header_image />
            <sys_id>95fc11615f1211001c9b2572f2b477c6</sys_id>
            <description>Services offered by different departments in the organization</description>
            <title>Departmental Services</title>
        </categories>
        <categories>
            <header_image />
            <sys_id>900682363731300054b6a3549dbe5d5f</sys_id>
            <description>Desktop computers for your work area.</description>
            <title>Desktops</title>
        </categories>
        <categories>
            <header_image />
            <sys_id>d2f716fccc611227a015a142fa0b262c1</sys_id>
            <description>Order new furniture, and fixtures, or request for furniture to be repaired. Cubicle modifications can also be ordered here.</description>
            <title>Furniture and Decor</title>
        </categories>
    </result>
</response>
Order from a variety of hardware to meet your business needs, including phones, tablets and laptops.

Request for cleaning services to be performed

Request for a shared office equipment to be repaired

Request for a standard change templates relating to network related changes: Adding new switches, upgrading IOS etc.

Office services such as printing, supplies requisition and document shipping and delivery.

End user peripherals such as mobile phone cases, dongles, and cables.
installation, providing different feature sets.</description>
</title>Printers</title>
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
<title>Printers</title>
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image />
</categories>
<header_image ../>
Service Catalog API - GET /sn_sc/servicecatalog/catalogs/{sys_id}/categories

This method retrieves a list of categories for a catalog.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/catalogs/{sys_id}/categories
Default URL: /api/sn_sc/servicecatalog/catalogs/{sys_id}/categories

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparam_top_level_only</td>
<td>Gets only those categories whose parent is a catalog.</td>
</tr>
<tr>
<td>sysparam_limit</td>
<td>Number of categories in the result.</td>
</tr>
<tr>
<td>sysparam_view</td>
<td>Specify the device to display the item (desktop, mobile, or both).</td>
</tr>
<tr>
<td>sysparam_offset</td>
<td>Offset to set the query window.</td>
</tr>
</tbody>
</table>
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Returns a list of categories in a catalog. If the query results in no categories then an empty array is returned.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates that the request is invalid and catalog sys_id is invalid.</td>
</tr>
<tr>
<td>500</td>
<td>Internal error.</td>
</tr>
</tbody>
</table>

Sample cURL request and response

```
curl "https://instance.service-now.com/api/sn_sc/v1/servicecatalog/catalogs/e0d08b13c3330100c8b837659bba8fb4/categories?sysparm_limit=10&sysparm_offset=12;rel="first" \ --request GET \ --header "Accept:application/json" \ --user 'admin':'admin'
```

```json
{
    "result": [
        {
            "title": "Desktops",
            "description": "Desktop computers for your work area.",
            "full_description": null,
            "icon": "",
            "header_icon": "",
            "homepage_image": "1bface31475211002ee987e8dee49095.iix",
            "sys_id": "900682363731300054b6a3549d0e5d5f"
        }
    ]
}```
ServiceNow

Kingston

Now Platform Custom Business Applications

},
{"title": "Furniture and Decor",
"description": "Order new furniture, and fixtures,
or request for\n\t\t\tfurniture to be repaired. Cubicle
modifications can also be ordered\n\t\t\there\n\t\t",
"full_description": null,
"icon": "",
"header_icon": "",
"homepage_image": "",
"sys_id": "d2f716fcc611227a015a142fa0b262c1"
},
{"title": "Hardware",
"description": "Order from a variety of hardware to meet
your business\n\t\t\tneeds, including phones, tablets and
laptops.",
"full_description": null,
"icon": "",
"header_icon": "",
"homepage_image": "c4b933e9471211002ee987e8dee49064.iix",
"sys_id": "d258b953c611227a0146101fb1be7c31"
},
{"title": "Janitorial",
"description": "Request for cleaning services to be
performed\n\t\t",
"full_description": null,
"icon": "",
"header_icon": "",
"homepage_image": "",
"sys_id": "d2f83003c611227a01a81bd5bd54b1aa"
},
{"title": "Laptops",
"description": "Laptop computers for mobile workers.",
"full_description": null,
"icon": "",
"header_icon": "",
"homepage_image": "ec5b0271475211002ee987e8dee49042.iix",
"sys_id": "59f586f23731300054b6a3549dbe5db7"
},
{"title": "Maintenance and Repair",
"description": "Request for a shared office equipment to
be repaired\n\t\t",
"full_description": null,
"icon": "",
"header_icon": "",
"homepage_image": "",
"sys_id": "d2f86388c611227a002209db6966d5ad"
},
{"title": "Mobiles",
"description": "Cell phones to meet your business needs.",
"full_description": null,
"icon": "",
"header_icon": "",
"homepage_image": "ff6b0271475211002ee987e8dee490d2.iix",
"sys_id": "d68eb4d637b1300054b6a3549dbe5db2"
},
{"title": "Network Standard Changes",
"description": "Standard change templates relating to
network related changes: Adding new switches, upgrading IOS
etc",
"full_description": null,
"icon": "",
"header_icon": "",
"homepage_image": "",
"sys_id": "abbcbbbf47700200e90d87e8dee49041"
© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States
and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.

1705


Sample Python request

```
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/v1/servicecatalog/catalogs/e0d08b13c33010c9b837659bba8fb4/categories?sysparm_limit=10&sysparm_offset=12;rel="next"'
url = 'https://instance.service-now.com/api/sn_sc/v1/servicecatalog/catalogs/e0d08b13c33010c9b837659bba8fb4/categories?sysparm_limit=10&sysparm_offset=20;rel="last"'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/json","Accept":"application/json"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)
```

None
Service Catalog API - GET /sn_sc/servicecatalog/categories/{sys_id}
This method retrieves all the information about a requested category.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/categories/{sys_id}
Default URL: /api/sn_sc/servicecatalog/categories/{sys_id}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_view</td>
<td>Defines the device that displays the Item:</td>
</tr>
<tr>
<td></td>
<td>· Desktop(desktop)</td>
</tr>
<tr>
<td></td>
<td>· Mobile(mobile)</td>
</tr>
<tr>
<td></td>
<td>· Desktop and Mobile(both)</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.
Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request has completed successfully and returns all the details of the category.</td>
</tr>
</tbody>
</table>
| 400         | Indicates that request is invalid for one of the following reasons:  
  - User does not have access to the category.  
  - Invalid sys_id. |
| 500         | Internal error occurred while executing the request. |

Response body values

Following are JSON or XML response parameters for a request completed successfully. The response consists of details of the category.

<table>
<thead>
<tr>
<th>Response parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>Title of the category.</td>
</tr>
<tr>
<td>sys_id</td>
<td>SysId of the category.</td>
</tr>
<tr>
<td>icon</td>
<td>The source of the small icon displayed beside the category name, when the category is listed as a subcategory.</td>
</tr>
<tr>
<td>header_icon</td>
<td>The source of the icon displayed beside the category header, when the category is listed at the top-level category.</td>
</tr>
<tr>
<td>description</td>
<td>Short description about the category.</td>
</tr>
<tr>
<td>full_description</td>
<td>Full description about the category.</td>
</tr>
<tr>
<td>homepage_image</td>
<td>The HomePageImage source of the category.</td>
</tr>
<tr>
<td>mobile_image</td>
<td>The image source used for the category in mobiles.</td>
</tr>
<tr>
<td>child_categories</td>
<td>The list of categories that belong to this category. Each object in the list gives the sys_id and title of the category that belong to this category.</td>
</tr>
</tbody>
</table>

Sample cURL request

```bash
curl "https://instance.service-now.com/api/sn_sc/servicecatalog/categories/b0fdfe01932002009ca87a75e57ffbe9" \
   --request GET \
   --header "Accept:application/json"
```
---

```json
{
  "result": {
    "title": "Standard Changes",
    "description": "Standard Change Template Library",
    "full_description": null,
    "icon": "",
    "header_icon": "",
    "homepage_image": "",
    "sys_id": "b0fdfb01932002009ca87a75e57ffbe9",
    "child_categories": [
    {
      "sys_id": "abbcbbbf47700200e90d87e8dee49041",
      "title": "Network Standard Changes"
    },
    {
      "sys_id": "b3ecbbbf47700200e90d87e8dee49081",
      "title": "Server Standard Changes"
    },
    {
      "sys_id": "00728916937002002dcef157b67ff6d",
      "title": "Template Management"
    }
    ]
  }
}
```

**Sample Python request**

```python
# Need to install requests package for python
# easy_install requests

import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/servicecatalog/categories/b0fdfb01932002009ca87a75e57ffbe9'

# Eg. User name='admin', Password='admin' for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type': 'application/json', 'Accept': 'application/xml'}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
```
Service Catalog API - GET /sn_sc/servicecatalog/items

This method retrieves a list of catalogs and a list of items for each catalog.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/items
Default URL: /api/sn_sc/servicecatalog/items

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_category</td>
<td>Specify the category sys_id of the item.</td>
</tr>
<tr>
<td>sysparm_type</td>
<td>Specify the type of item. For example, Record Producer, Order Guide. This field is used by the item-search to search specified item types.</td>
</tr>
<tr>
<td>sysparm_limit</td>
<td>Specify the number of entities in the response. This value applies to categories and items.</td>
</tr>
<tr>
<td>sysparm_text</td>
<td>Specify query text for search support. (Required)</td>
</tr>
<tr>
<td>sysparm_offset</td>
<td>Specify the offset used for pagination.</td>
</tr>
<tr>
<td>sysparm_catalog</td>
<td>Specify the catalog sys_id of the item.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sysparm_view</td>
<td>Defines the device that displays the Item:</td>
</tr>
<tr>
<td></td>
<td>· Desktop (desktop)</td>
</tr>
<tr>
<td></td>
<td>· Mobile (mobile)</td>
</tr>
<tr>
<td></td>
<td>· Desktop and Mobile (both)</td>
</tr>
</tbody>
</table>

**Headers**

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

**Request headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Response headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Status codes**

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

**Status codes**

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Returns a list of catalogs and items available. If a valid query results in no items, then the response body contains only an empty result array.</td>
</tr>
<tr>
<td>500</td>
<td>Internal error.</td>
</tr>
</tbody>
</table>

**Sample cURL request**

```bash
curl "https://instance.service-now.com/api/sn_sc/servicecatalog/items?sysparm_limit=2" \
--request GET \
--header "Accept:application/json" \
--user 'admin':'admin'

{ "result": [  
```
Sample Python request

# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/v1/servicecatalog/items?sysparm_limit=10'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers

Service Catalog API - GET /sn_sc/servicecatalog/items/{sys_id}

This method retrieves the catalog item with the specified sys_id.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/items/{sys_id}
Default URL: /api/sn_sc/servicecatalog/items/{sys_id}
Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_view</td>
<td>Defines the device that displays the Item:</td>
</tr>
<tr>
<td></td>
<td>· Desktop(desktop)</td>
</tr>
<tr>
<td></td>
<td>· Mobile(mobile)</td>
</tr>
<tr>
<td></td>
<td>· Desktop and Mobile(both)</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request has completed successfully and returns the details of the catalog item.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates that request is invalid for one of the following reasons:</td>
</tr>
<tr>
<td></td>
<td>· User does not have access to the catalog item.</td>
</tr>
<tr>
<td></td>
<td>· Invalid sys_id.</td>
</tr>
</tbody>
</table>
### Request body

The API accepts these JSON or XML elements in the request body.

<table>
<thead>
<tr>
<th>Element accepted in the request body</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

### Response body

The API returns these JSON or XML elements in the response body to describe the catalog item.

<table>
<thead>
<tr>
<th>Elements returned in the response body</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
<tr>
<td>sys_id</td>
</tr>
<tr>
<td>type</td>
</tr>
<tr>
<td>short_description</td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>icon</td>
</tr>
<tr>
<td>category</td>
</tr>
<tr>
<td>Element</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>show_price</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>catalogs</td>
</tr>
<tr>
<td>variables</td>
</tr>
<tr>
<td>ui_policy</td>
</tr>
<tr>
<td>client_script</td>
</tr>
<tr>
<td>data_lookup</td>
</tr>
</tbody>
</table>

**Sample cURL request**

curl "https://instance.service-now.com/api/sn_sc/servicecatalog/items/0cf7477237153000158bbfc8bcbe5dec" \
--request GET \
--header "Accept:application/json" \
--user 'admin':'admin'

"result": {
  "short_description": "Google Nexus 7",
  "icon": "images/service_catalog/generic_small.gifx",
  "description": "\n<h3>Google Nexus 7 Black Wi-Fi 16GB Tablet, Model NEXUS7ASUS1B16</h3>\n<p style="margin-bottom: 0px; \""><h4>Key Features:</h4><li>The world’s 1st Android™ 4.1 Jelly Bean tablet</li><li>World’s first 7" quad-core tablet delivers an advanced multimedia experience with up to 9.5* hours of battery life</li><li>World’s best NFC experience on a tablet with a textured tactile design for enhanced comfort measuring just 10.45mm thin and 340g light</li><li>ASUS TruVivid technology with Corning® Fit Glass for improved color clarity and scratch resistance</li><li>178 wide-viewing angle IPS display ensures unrivaled visual acuity</li>\n"show_price": true,
"recurring_price": "$10.00",
"type": "catalog_item",
"local_currency": "USD",
"sys_id": "0cf7477237153000158bbfc8bcbe5dec",
"recurring_price_currency": "USD",
"localized_price": "$199.99",
"price": "$199.99",
"catalogs": [
Sample Python request

```python
# Need to install requests package for python
# easy_install requests

import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/servicecatalog/items/0cf7477237153000158bbfc8bcbe5dec'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = ({"Content-Type":"application/xml","Accept":"application/xml"})

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.json())
exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

<?xml version="1.0" encoding="UTF-8"?>
<response>
  <result>
    <short_description>Google Nexus 7</short_description>
    <icon>images/service_catalog/generic_small.gifx</icon>
    <description>&lt;h3&gt;Google Nexus 7 Black Wi-Fi 16GB Tablet, Model NEXUS7ASUS1B16&lt;/h3&gt;
      &amp;lt;p style="margin-bottom: 0px;"&gt;&amp;nbsp;&amp;lt;b&amp;gt;Key Features:&amp;lt;/b&amp;gt;&amp;lt;/p&amp;gt;
      &amp;lt;li&amp;gt;The world&amp;amp;#39;s 1st Android™ 4.1 Jelly Bean tablet&amp;lt;/li&amp;gt;
      &amp;lt;li&amp;gt;World’s first 7” quad-core tablet delivers an advanced multimedia experience with up to 9.5* hours of battery life&amp;lt;/li&amp;gt;
      &amp;lt;li&amp;gt;World&amp;amp;#39;s best NFC experience on a tablet with a textured tactile design for enhanced comfort measuring just 10.45mm thin and 340g light&amp;lt;/li&amp;gt;
      &amp;lt;li&amp;gt;ASUS TruVivid technology with Corning® Fit Glass for improved color clarity and scratch resistance&amp;lt;/li&amp;gt;
      &amp;lt;li&amp;gt;178 wide-viewing angle IPS display ensures unrivaled visual acuity&amp;lt;/li&amp;gt;&lt;/description&gt;
    <show_price />
    <recurring_price>$10.00</recurring_price>
    <type>catalog_item</type>
    <local_currency>USD</local_currency>
    <sys_id>0cf7477237153000158bbfc8bcbe5dec</sys_id>
    <recurring_price_currency>USD</recurring_price_currency>
    <localized_price>$199.99</localized_price>
    <client_script />
    <price>$199.99</price>
    <catalogs>
      <sys_id>e0d08b13c3330100c8b837659bba8fb4</sys_id>
      &lt;title&gt;Service Catalog&lt;/title&gt;
    </catalogs>
    <recurring_frequency>Weekly</recurring_frequency>
    <name&gt;Google Nexus 7&lt;/name&gt;
    <localized_recurring_price>$10.00</localized_recurring_price>
    <categories>
      <sys_id>d258b953c611227a0146101fb1be7c31</sys_id>
      &lt;active />&lt;/category&gt;
      &lt;sys_id>b06546f23731300054b6a3549dbe5dd8</sys_id>
      &lt;active />&lt;/category&gt;
      &lt;title&gt;Tablets&lt;/title&gt;
    </categories>
    &lt;title&gt;Hardware&lt;/title&gt;
    &lt;categories&gt;
      &lt;sys_id>b06546f23731300054b6a3549dbe5dd8</sys_id>
      &lt;title&gt;Tablets&lt;/title&gt;
    &lt;/category&gt;
    &lt;price_currency&gt;USD&lt;/price_currency&gt;
  </result>
</response>
Service Catalog API - GET /sn_sc/servicecatalog/wishlist

This method retrieves list of items from the user's wish list.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/wishlist
Default URL: /api/sn_sc/servicecatalog/wishlist

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request completed successfully. Returns the user's list of wish list items.</td>
</tr>
</tbody>
</table>
Sample cURL request

curl "http://localhost/api/sn_sc/servicecatalog/wishlist" \
   --request GET \
   --header "Accept:application/json" \
   --user 'admin':'admin'

{
   "result": {
      "cart_id": "02a559a7c3b02200d68d3b0ac3d3ae5d",
      "items": [
         {
            "catalog_item_id": "04b7e94b4f7b4200086eeed18110c7fd",
            "variables": {
               "Adobe Photoshop": "",
               "Adobe Acrobat": "",
               "Optional Software": "",
               "Additional software requirements": ""
            },
            "quantity": "1",
            "localized_price": "$1,100.00",
            "price": "$1,100.00",
            "recurring_frequency": "Annually",
            "localized_recurring_price": "$100.00",
            "recurring_price": "$100.00",
            "item_name": "Standard Laptop &",
            "cart_item_id": "d31be364c3012200d68d3b0ac3d3ae5d",
            "delivery_time": "5 Days"
         }
      ]
   }
}

Sample Python request

# Need to install requests package for python
# easy_install requests

import requests

# Set the request parameters
url = 'http://localhost/api/sn_sc/servicecatalog/wishlist'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type': 'application/json', 'Accept': 'application/json'}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd),
headers=headers)

    # Check for HTTP codes other than 200
    if response.status_code != 200:
        print('Status:', response.status_code, 'Headers:',
        response.headers, 'Error Response:',response.json())
        exit()

    # Decode the JSON response into a dictionary and use the
data
    data = response.json()
    print(data)

<?xml version="1.0" encoding="UTF-8"?>
<response>
  <result>
    <cart_id>02a559a7c3b02200d68d3b0ac3d3ae5d</cart_id>
    <items>
      <catalog_item_id>04b7e94b4f7b4200086eed18110c7fd</catalog_item_id>
      <variables>
        <Adobe Photoshop></Adobe Photoshop>
        <Adobe Acrobat></Adobe Acrobat>
        <Optional Software></Optional Software>
        <Additional software requirements></Additional software requirements>
      </variables>
      <quantity>1</quantity>
      <localized_price>$1,100.00</localized_price>
      <price>$1,100.00</price>
      <recurring_frequency>Annually</recurring_frequency>
      <localized_recurring_price>$100.00</localized_recurring_price>
      <recurring_price>$100.00</recurring_price>
      <item_name>Standard Laptop &amp;</item_name>
      <cart_item_id>d31be364c3012200d68d3b0ac3d3ae5f</cart_item_id>
      <delivery_time>5 Days</delivery_time>
    </items>
  </result>
</response>

Service Catalog API - GET /sn_sc/servicecatalog/wishlist/{cart_item_id}
This method retrieves the details of the specified item stored in the wish list cart.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/wishlist/{cart_item_id}
Default URL: /api/sn_sc/servicecatalog/wishlist/{cart_item_id}
Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request completed successfully. Returns details of the specific item in the wish list.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates a failed request. Incorrect cart item id or the item does not exist in the wish list.</td>
</tr>
</tbody>
</table>

Sample cURL request

curl "http://localhost/api/sn_sc/servicecatalog/wishlist/d31be364c3012200d68d3b0ac3d3aecf" \
   --request GET \
   --header "Accept:application/json" \
   --user 'admin':'admin'

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'http://localhost/api/sn_sc/servicecatalog/wishlist/d31be364c3012200d68d3b0ac3d3aefc'
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type': 'application/json', 'Accept': 'application/json'}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <result>
    <catalog_item_id>04b7e94b4f7b4200086eeed18110c7fd</catalog_item_id>
    <quantity>1</quantity>
    <variables>
      "Adobe Photoshop": 
      "Adobe Acrobat": 
      "Optional Software": 
      "Additional software requirements": 
    
```
Service Catalog API - POST /sn_sc/servicecatalog/cart/{cart_item_id}

This method edits and updates any item in the cart.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/cart/{cart_item_id}
Default URL: /api/sn_sc/servicecatalog/cart/{cart_item_id}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Indicates that the request completed successfully and that the item is added to the cart.</td>
</tr>
</tbody>
</table>
| 400         | Following three errors can occur:  

- **Invalid quantity value**: Indicates that either the `sysparm_quantity` parameter is not provided or contains invalid value.  
- **Security constraints prevent ordering of Item**: Indicates that either the Item ID specified in the path parameters is either invalid or the user does not have access to the item.  
- **Mandatory Variables are required**: Indicates that one or more of the mandatory variable values is not provided in the request. |

Request and Response body

**Request body parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sysparm_quantity</code></td>
<td>Specifies the quantity of the item to update the cart (required parameter).</td>
</tr>
<tr>
<td><code>variables</code></td>
<td>Object consisting of all variable names and their corresponding valid values. Provide values for all the mandatory variables.</td>
</tr>
</tbody>
</table>

**Response body parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cart_id</code></td>
<td>Specifies the sys_id of the cart.</td>
</tr>
<tr>
<td><code>subtotal_price</code></td>
<td>Specifies the subtotal of the cart.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>subtotal_recurring_frequency</td>
<td>Specifies the recurring frequency subtotal of the cart.</td>
</tr>
<tr>
<td>subtotal_recurring_price</td>
<td>Specifies the recurring price subtotal of the cart.</td>
</tr>
<tr>
<td>total_title</td>
<td>Specifies the title for total field on page.</td>
</tr>
<tr>
<td>show_subtotal_price</td>
<td>Specifies Boolean value to show or not show subtotal price.</td>
</tr>
<tr>
<td>subtotal_title</td>
<td>Specifies the title for subtotal field on page.</td>
</tr>
<tr>
<td>monthly, weekly, daily</td>
<td>Array of item details classified by recurring frequencies of the items in the cart.</td>
</tr>
<tr>
<td></td>
<td>- subtotal_price: Specifies the subtotal of the frequency block.</td>
</tr>
<tr>
<td></td>
<td>- subtotal_recurring_frequency: Specifies the recurring frequency subtotal of the frequency block.</td>
</tr>
<tr>
<td></td>
<td>- subtotal_recurring_price: Specifies the recurring price subtotal of the frequency block.</td>
</tr>
<tr>
<td></td>
<td>- total_title: Title for total field on the frequency block.</td>
</tr>
<tr>
<td>items</td>
<td>Array consisting of details of all the items in the cart.</td>
</tr>
<tr>
<td></td>
<td>- catalog_item_id: sys_id of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- variables: Object consisting of all the variable names and corresponding values for the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- quantity: Quantity of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- localized_price: Price of the item in local currency.</td>
</tr>
<tr>
<td></td>
<td>- price: Price of the item.</td>
</tr>
<tr>
<td></td>
<td>- localized_recurring_price: Recurring price of the item in local currency.</td>
</tr>
<tr>
<td></td>
<td>- recurring_price: Recurring price of the item.</td>
</tr>
<tr>
<td></td>
<td>- recurring_frequency: Recurring frequency of the item.</td>
</tr>
<tr>
<td></td>
<td>- item_name: Name of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- cart_item_id: sys_id of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- delivery_time: Delivery time of the item.</td>
</tr>
</tbody>
</table>

**Sample cURL request**

```bash
curl "https://instance.service-now.com/api/sn_sc/v1/servicecatalog/cart/183f0297db21120064301150f0b8f5fe" \
--request POST \
--header "Accept:application/json" \
```
--user 'admin': 'admin'

```json
{
    "result":{
        "cart_id": "616ebdb8db71120076861150f0b8f552",
        "subtotal_price": "$9,599.88",
        "subtotal_recurring_frequency": "Monthly",
        "subtotal_recurring_price": "$348.00",
        "total_title": "Total",
        "monthly":{
            "subtotal_price": "$9,599.88",
            "subtotal_recurring_frequency": "Monthly",
            "subtotal_recurring_price": "$348.00",
            "total_title": "Total",
            "items":{
                "catalog_item_id": "0d08837237153000158bbfc8bcbe5d02",
                "variables":{
                    "Monthly data allowance": "500MB",
                    "Storage": "16GB",
                    "Color": "Black",
                    "Allocated carrier": "AT & T Mobility",
                    "Contract duration": "18 Months"
                },
                "quantity": "12",
                "localized_price": "$799.99",
                "price": "$799.99",
                "recurring_frequency": "Monthly",
                "localized_recurring_price": "$30.00",
                "recurring_price": "$29.00",
                "item_name": "Apple iPhone 5",
                "cart_item_id": "a21ffdb0db71120076861150f0b8f560",
                "delivery_time": "2 Days"  // Check this
            }
        },
        "show_subtotal_price": "true",
        "subtotal_title": "Subtotal"
    }
}
```
headers = {"Content-Type": "application/xml", "Accept": "application/xml"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 201
if response.status_code != 201:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <result>
    <cart_id>616ebdb8db71120076861150f0b8f552</cart_id>
    <subtotal_price>$9,599.88</subtotal_price>
    <subtotal_recurring_frequency>Monthly</subtotal_recurring_frequency>
    <subtotal_recurring_price>$348.00</subtotal_recurring_price>
    <total_title>Total</total_title>
    <monthly>
      <subtotal_price>$9,599.88</subtotal_price>
      <subtotal_recurring_frequency>Monthly</subtotal_recurring_frequency>
      <subtotal_recurring_price>$348.00</subtotal_recurring_price>
      <total_title>Total</total_title>
    </monthly>
    <items>
      <catalog_item_id>0d08837237153000158bbfc8bcbe5d02</catalog_item_id>
      <variables>
        <Monthly data allowance>500MB</Monthly data allowance>
        <Storage>16GB</Storage>
        <Color>Black</Color>
        <Allocated carrier>AT & T Mobility</Allocated carrier>
        <Contract duration>18 Months</Contract duration>
      </variables>
      <quantity>12</quantity>
      <localized_price>$799.99</localized_price>
      <price>$799.99</price>
      <recurring_frequency>Monthly</recurring_frequency>
      <localized_recurring_price>$30.00</localized_recurring_price>
      <recurring_price>$29.00</recurring_price>
      <item_name>Apple iPhone 5</item_name>
      <cart_item_id>a21ffdb0db71120076861150f0b8f560</cart_item_id>
      <delivery_time>2 Days</delivery_time>
    </items>
  </result>
  <show_subtotal_price>true</show_subtotal_price>
  <subtotal_title>Subtotal</subtotal_title>
</root>
```
Service Catalog API - POST /sn_sc/servicecatalog/cart/checkout

This method retrieves the checkout cart details based on the two-step checkout process enabled or disabled. If the user enables two-step checkout, the method returns cart order status and all the information required for two-step checkout. If the user disables two-step checkout, the method checks out the cart and returns the request number and request order ID.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/cart/checkout
Default URL: /api/sn_sc/servicecatalog/cart/checkout

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.
Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request completed successfully. Returns the cart status with the checkout status of the items in the cart.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates that the cart is empty and cannot check out.</td>
</tr>
</tbody>
</table>

Request body

The API accepts these JSON or XML elements in the request body.

Elements accepted in the request body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response body

If two-step checkout is enabled, the API returns these JSON or XML elements in the response body to describe the order status page.

Elements returned in the response body for two-step checkout

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cart_id</td>
<td>Specifies the sys_id of the cart.</td>
</tr>
<tr>
<td>subtotal_price</td>
<td>Specifies the subtotal of the cart.</td>
</tr>
<tr>
<td>subtotal_recurring_frequency</td>
<td>Specifies the recurring frequency subtotal of the cart.</td>
</tr>
<tr>
<td>subtotal_recurring_price</td>
<td>Specifies the recurring price subtotal of the cart.</td>
</tr>
<tr>
<td>total_title</td>
<td>Specifies the title for total field on page.</td>
</tr>
<tr>
<td>subtotal_price</td>
<td>Specifies the subtotal of the current frequency block.</td>
</tr>
<tr>
<td>subtotal_recurring_frequency</td>
<td>Specifies the recurring frequency subtotal of the current frequency block.</td>
</tr>
<tr>
<td>subtotal_recurring_price</td>
<td>Specifies the recurring price subtotal of the current frequency block.</td>
</tr>
<tr>
<td>total_title</td>
<td>Specifies the title for total field on the current frequency block.</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>items</td>
<td>Array consisting of details of all the items in the cart.</td>
</tr>
<tr>
<td></td>
<td>- catalog_item_id: sys_id of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- variables: Object consisting of all the variable names and corresponding values for the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- quantity: Quantity of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- localized_price: Price of the item in local currency.</td>
</tr>
<tr>
<td></td>
<td>- price: Price of the item.</td>
</tr>
<tr>
<td></td>
<td>- localized_recurring_price: Recurring price of the item in local currency.</td>
</tr>
<tr>
<td></td>
<td>- recurring_price: Recurring price of the item.</td>
</tr>
<tr>
<td></td>
<td>- recurring_frequency: Recurring frequency of the item.</td>
</tr>
<tr>
<td></td>
<td>- item_name: Name of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- cart_item_id: sys_id of the item in the cart.</td>
</tr>
<tr>
<td></td>
<td>- delivery_time: Delivery time of the item.</td>
</tr>
</tbody>
</table>

### Sample cURL request

```
curl "https://instance.service-now.com/api/sn_sc/servicecatalog/cart/checkout" \
--request POST \
--header "Accept:application/json" \
--user 'admin':'admin'
```

<!-- This example response is returned when two-step checkout is disabled -->

```json
{
   "result": {
      "request_number": "REQ0010009",
      "request_id": "6e240197db21120064301150f0b8f5be"
   }
}
```

### Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/servicecatalog/cart/checkout'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
```
headers = {"Content-Type":"application/json","Accept":"application/json"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

<!-- This example response is returned when two-step checkout is enabled -->
<?xml version="1.0" encoding="UTF-8" ?>
<root>
    <result>
        <cart_id>616ebdb8db71120076861150f0b8f552</cart_id>
        <subtotal_price>$9,599.88</subtotal_price>
        <subtotal_recurring_frequency>Monthly</subtotal_recurring_frequency>
        <subtotal_recurring_price>$348.00</subtotal_recurring_price>
        <total_title>Total</total_title>
        <delivery_address>Brasilia, Brasil</delivery_address>
        <special_instructions></special_instructions>
        <requested_for_user>System Administrator</requested_for_user>
        <monthly>
            <subtotal_price>$9,599.88</subtotal_price>
            <subtotal_recurring_frequency>Monthly</subtotal_recurring_frequency>
            <subtotal_recurring_price>$348.00</subtotal_recurring_price>
            <total_title>Total</total_title>
        </monthly>
        <items>
            <catalog_item_id>0d08837237153000158bbfc0bcbe5d02</catalog_item_id>
            <variables>
                <Monthly data allowance>500MB</Monthly data allowance>
                <Storage>16GB</Storage>
                <Color>Black</Color>
                <Allocated carrier>AT & T Mobility</Allocated carrier>
                <Contract duration>18 Months</Contract duration>
            </variables>
            <quantity>12</quantity>
            <localized_price>$799.99</localized_price>
            <price>$799.99</price>
            <recurring_frequency>Monthly</recurring_frequency>
            <localized_recurring_price>$30.00</localized_recurring_price>
        </items>
    </result>
</root>
Service Catalog API - POST /sn_sc/servicecatalog/cart/submit_order

This method checks out the user cart, whether two-step parameter is enabled or disabled.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/cart/submit_order
Default URL: /api/sn_sc/servicecatalog/cart/submit_order

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request completed successfully and the items in the cart are successfully checked out.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates that the user cart is empty.</td>
</tr>
</tbody>
</table>

Request body

The API accepts these JSON or XML elements in the request body.

<table>
<thead>
<tr>
<th>Elements accepted in the request body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

Response body

The API returns these JSON or XML elements in the response body.

<table>
<thead>
<tr>
<th>Response body values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>request_number</td>
</tr>
<tr>
<td>request_id</td>
</tr>
</tbody>
</table>

Sample cURL request

```bash
curl "https://instance.service-now.com/api/sn_sc/v1/servicecatalog/cart/submit_order" 
--request POST 
--header "Accept:application/json" 
--user 'admin':'admin'

{
   "result": {
      "request_number": "REQ0010010",
      "request_id": "f89ed597db21120064301150f0b8f5ff"
   }
}
```
Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/v1/servicecatalog/cart/submit_order'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/xml","Accept":"application/xml"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)
```

Service Catalog API - POST /sn_sc/servicecatalog/items/{sys_id}/add_to_cart

Adds the specified item to the cart of the current user.

**URL format**

Versioned URL: /api/sn_sc/{version}/servicecatalog/items/{sys_id}/add_to_cart

Default URL: /api/sn_sc/servicecatalog/items/{sys_id}/add_to_cart
Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Optional. Version of the endpoint to access. For example, ‘v1’ or ‘v2’. Only specify this value to use an endpoint version other than the “latest”.</td>
</tr>
<tr>
<td>sys_id</td>
<td>Unique sys_id of the item to add to the current cart.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Data format of the response body. Supported types: application/json or application/xml. Default: application/json</td>
</tr>
<tr>
<td>Content-Type</td>
<td>Data format of the request body. Supported types: application/json or application/xml. Default: application/json</td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

Endpoint status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request completed successfully and the items have been added to the cart. The response contains the details of the items currently in the cart.</td>
</tr>
</tbody>
</table>
## Status code

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
</table>
| 400         | The following errors can occur:  
  - **Invalid quantity value**: indicates that either the `sysparm_quantity` parameter is not provided or contains an invalid value.  
  - **Security constraints prevent ordering of item**: indicates that either the cart item `sys_id` specified in the path parameters is invalid or the user does not have access to the item.  
  - **Mandatory Variables are required**: indicates that one or more of the mandatory variables is not provided in the request. |

## Request body

The endpoint accepts these JSON or XML elements in the request body.

### Elements required in the request body

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_quantity</td>
<td>Required. Number of the item to add to the cart. Cannot be a negative number.</td>
</tr>
<tr>
<td>variables</td>
<td>Name/value pairs of all mandatory cart item variables. Mandatory variables are defined on the associated form.</td>
</tr>
</tbody>
</table>

## Response body

The endpoint returns these JSON or XML elements in the response body.

### Elements returned in the response body

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cart_id</td>
<td>Unique cart sys_id.</td>
</tr>
<tr>
<td>items</td>
<td>Details of all of the items in the cart.</td>
</tr>
<tr>
<td>items.cart_item_id</td>
<td>Unique sys_id of the item in the cart.</td>
</tr>
<tr>
<td>items.catalog_item_id</td>
<td>Unique catalog sys_id of the item in the cart.</td>
</tr>
<tr>
<td>items.localized_price</td>
<td>Price of the item, in the user's local currency.</td>
</tr>
<tr>
<td>items.localized_recurring_price</td>
<td>Recurring price of the item, in the user's local currency.</td>
</tr>
<tr>
<td>items.item_name</td>
<td>Name of the item in the cart.</td>
</tr>
<tr>
<td>items.price</td>
<td>Base price of the item.</td>
</tr>
<tr>
<td>items.quantity</td>
<td>Quantity of the item in the cart.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>items.recurring_frequency</td>
<td>Frequency at which the recurring price is applied.</td>
</tr>
<tr>
<td>items.recurring_price</td>
<td>Recurring price of the item.</td>
</tr>
<tr>
<td>subtotal</td>
<td>Subtotal of the items in the cart.</td>
</tr>
</tbody>
</table>

Sample cURL request and response

```bash
curl "https://instance.service-now.com/api/sn_sc/servicecatalog/items/0d08837237153000158bbfc8bcbe5d02/add_to_cart" \
--request POST \ 
--header "Accept:application/json" \ 
--data "{'sysparm_quantity': '1', 'variables': {'acrobat': 'true'}}" \ 
--user 'username':'password'
```

```json
{
    "result":{
        "cart_id":"0951c597db21120064301150f0b8f5cf",
        "subtotal": "$1,599.98",
        "items": [
            {
                "catalog_item_id":"0d08837237153000158bbfc8bcbe5d02",
                "quantity": "2",
                "localized_price": "$799.99",
                "price": "$799.99",
                "recurring_frequency": "Monthly",
                "localized_recurring_price": "$30.00",
                "recurring_price": "$29.00",
                "item_name": "Apple iPhone 5",
                "cart_item_id": "ac61c597db21120064301150f0b8f5f9"
            }
        ]
    }
}
```

Sample Python request and response

```python
# Need to install requests package for python
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/servicecatalog/items/0d08837237153000158bbfc8bcbe5d02/add_to_cart'

# Eg. User name="username", Password="password" for this code sample.
user = 'username'
pwd = 'password'

# Set proper headers
```
headers = {"Content-Type":"application/json","Accept":"application/json"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd), headers=headers, data="{"sysparm_quantity": \"2\"}")

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

{
    "result": {
        "cart_id": "0830db8413a56300397533e2e144b0ba",
        "subtotal": "$799.00",
        "items": [
            {
                "catalog_item_id": "07f1666b0bf00300eba42da0d5673ab0",
                "quantity": "1",
                "localized_price": "$799.00",
                "price": "$799.00",
                "recurring_frequency": null,
                "localized_recurring_price": "$0.00",
                "recurring_price": "$0.00",
                "item_name": "Apple iPhone7",
                "cart_item_id": "7d138fa21329e300027879d96144b065"
            }
        ]
    }
}

Service Catalog API - POST /sn_sc/servicecatalog/items/{sys_id}/add_to_wishlist

This method adds an item to the wish list cart.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/items/{sys_id}/add_to_wishlist
Default URL: /api/sn_sc/servicecatalog/items/{sys_id}/add_to_wishlist

Supported request parameters

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request completed successfully. The item is added to wish list.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates a failed request. Invalid request data or the user cannot add item to the wish list.</td>
</tr>
</tbody>
</table>

Sample cURL request

```
curl "http://localhost/api/sn_sc/servicecatalog/items/04b7e94b4f7b4200086eeed18110c7fd/add_to_wishlist" 
    --request POST 
    --header "Accept:application/json" 
    --header "Content-Type:application/json" 
    --data "{"sysparm_quantity":"1"}" 
    --user 'admin':'admin'
```

```json
{
    "result": {,
    "cart_id": "02a559a7c3b02200d68d3b0ac3d3ae5d",
    "items": [,
    {
    "catalog_item_id": "04b7e94b4f7b4200086eeed18110c7fd",
    "variables": {
    "Adobe Photoshop": "",
    "Adobe Acrobat": "",
    "Optional Software": ""
    }
    }
    ]
}
```
"Additional software requirements": "",
"quantity": "1",
"localized_price": "$1,100.00",
"price": "$1,100.00",
"recurring_frequency": "Annually",
"localized_recurring_price": "$100.00",
"recurring_price": "$100.00",
"item_name": "Standard Laptop &",
"cart_item_id": "d31be364c3012200d68d3b0ac3d3ae5d",
"delivery_time": "5 Days"}
]
]
]

Sample Python request

# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'http://localhost/api/sn_sc/servicecatalog/items/04b7e94b4f7b4200086eeed18110c7fd/add_to_wishlist'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/json","Accept":"application/json"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd), headers=headers, data="{"sysparm_quantity":"1"}"

# Check for HTTP codes other than 200
if response.status_code != 200:
  print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
  exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

<?xml version="1.0" encoding="UTF-8"?>
<response>
  <result>
    <cart_id>02a559a7c3b02200d68d3b0ac3d3ae5d</cart_id>
    <items>
      <variables>
        <Adobe Photoshop></Adobe Photoshop>
        <Adobe Acrobat></Adobe Acrobat>
        <Optional Software></Optional Software>
        <Additional software requirements></Additional software requirements>
Service Catalog API - POST /sn_sc/servicecatalog/items/{sys_id}/checkout_guide

This method retrieves an array of contents requested for checkout.

**URL format**

Versioned URL: /api/sn_sc/v1/servicecatalog/items/{sys_id}/checkout_guide
Default URL: /api/sn_sc/servicecatalog/items/{sys_id}/checkout_guide

**Supported request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Headers**

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

**Response headers**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request has completed successfully.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Is not two-step</strong>: Items are added to cart and cart contents are returned.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Is two-step</strong>: Returns the request number and request sys_id after checkout.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates that the request is invalid. Could be due to one of the following reasons:</td>
</tr>
<tr>
<td></td>
<td>- One or more items sent do not exist.</td>
</tr>
<tr>
<td></td>
<td>- User does not have access to one or more of the items.</td>
</tr>
<tr>
<td></td>
<td>- Mandatory variables of one or more items have not been answered. (Only variables defined as mandatory are honored.)</td>
</tr>
<tr>
<td></td>
<td>- Quantity value is sent and is not a positive integer.</td>
</tr>
<tr>
<td>500</td>
<td>Service error. Internal error during checkout.</td>
</tr>
</tbody>
</table>

Request body

The API requires these JSON or XML elements in the request body to specify a checkout guide.

Elements required in the request body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Array of items requested for checkout. Each item is a JSON object with the following parameters:</td>
</tr>
<tr>
<td></td>
<td>- <strong>sys_id</strong>: ID of the item, which is a required parameter.</td>
</tr>
<tr>
<td></td>
<td>- <strong>variables</strong>: JSON object of variable name value pairs.</td>
</tr>
<tr>
<td></td>
<td>- <strong>sysparam_quantity</strong>: Quantity of item. If not specified, the default value is 1.</td>
</tr>
</tbody>
</table>
Response body

If two-step verification is false, the API returns these JSON or XML elements in the response body to describe the order status.

Elements returned in the response body when two-step checkout is false

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cart_id</td>
<td>Specifies the sys_id of the cart.</td>
</tr>
<tr>
<td>subtotal_price</td>
<td>Specifies the subtotal of the cart.</td>
</tr>
<tr>
<td>subtotal_recurring_frequency</td>
<td>Specifies the recurring frequency subtotal of the cart.</td>
</tr>
<tr>
<td>subtotal_recurring_price</td>
<td>Specifies the recurring price subtotal of the cart.</td>
</tr>
<tr>
<td>total_title</td>
<td>Specifies the title for total field on page.</td>
</tr>
<tr>
<td>subtotal_price</td>
<td>Specifies the subtotal of the current frequency block.</td>
</tr>
<tr>
<td>subtotal_recurring_frequency</td>
<td>Specifies the recurring frequency subtotal of the current frequency block.</td>
</tr>
<tr>
<td>subtotal_recurring_price</td>
<td>Specifies the recurring price subtotal of the current frequency block.</td>
</tr>
<tr>
<td>total_title</td>
<td>Specifies the title for total field on the current frequency block.</td>
</tr>
<tr>
<td>items</td>
<td>Array consisting of details of all the items in the cart.</td>
</tr>
<tr>
<td>· catalog_item_id:</td>
<td>sys_id of the item in the cart.</td>
</tr>
<tr>
<td>· variables:</td>
<td>Object consisting of all the variable names and corresponding values for the item in the cart.</td>
</tr>
<tr>
<td>· quantity:</td>
<td>Quantity of the item in the cart.</td>
</tr>
<tr>
<td>· localized_price:</td>
<td>Price of the item in local currency.</td>
</tr>
<tr>
<td>· price:</td>
<td>Price of the item.</td>
</tr>
<tr>
<td>· localized_recurring_price:</td>
<td>Recurring price of the item in local currency.</td>
</tr>
<tr>
<td>· recurring_price:</td>
<td>Recurring price of the item.</td>
</tr>
<tr>
<td>· recurring_frequency:</td>
<td>Recurring frequency of the item.</td>
</tr>
<tr>
<td>· item_name:</td>
<td>Name of the item in the cart.</td>
</tr>
<tr>
<td>· cart_item_id:</td>
<td>sys_id of the item in the cart.</td>
</tr>
<tr>
<td>· delivery_time:</td>
<td>Delivery time of the item.</td>
</tr>
</tbody>
</table>

If two-step verification is true, the API returns these JSON or XML elements in the response body to describe the request.
Elements returned in the response body when two-step checkout is true

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request_number</td>
<td>Specifies the request record number.</td>
</tr>
<tr>
<td>request_id</td>
<td>Specifies the sys_id of the request.</td>
</tr>
</tbody>
</table>

Sample cURL request

```
curl "https://instance.service-now.com/api/sn_sc/v1/servicecatalog/items/6690750f4f7b4200086eeed18110c761/checkout_guide" \
--request POST \
--header "Accept:application/json" \
--header "Content-Type:application/json" \
--data "{"items":[
  {"sys_id":"04b7e94b4f7b4200086eeed18110c7fd","variables":{
    "acrobat":"true",
    "Additional_software_requirements":"MS Office 2007"},
  "sysparm_quantity":"3"},
  {"sys_id":"e1be6dcb4f7b4200086eeed18110c74c"},
  {"sys_id":"186d917a6fab7980575967d6bb3ee4f2","variables":{
    "new_email":"abel.tuter@example.com"}
  },
  {"sys_id":"8b3ae7fedc1be1004e5e5c08239e522b"}
},
{"sys_id":"962967674ff38200086eeed18110c7e7","variables":{
  "ergonomic_office":"Office chair should have lumbar support"}}]"

--user 'admin':'admin'
```

<!--Case-1: if two-step is false:-->
```
{ "result": {  "cart_id": "0d8c7a32c3211200d67d3b0ac3d3aeba",  "subtotal_price": "$3,350.00",  "subtotal_recuring_frequency": "",  "subtotal_recuring_price": "$300.00",  "total_title": "Total",  "monthly": {  "subtotal_price": "$50.00",  "subtotal_recuring_frequency": "Monthly",  "subtotal_recuring_price": "$0.00",  "total_title": "Total",  "items": [  {  "catalog_item_id": "90af095bcd38798071a208d71d1b64f",```
"variables": {},
"quantity": "1",
"localized_price": "$50.00",
"price": "$50.00",
"recurring_frequency": "Monthly",
"localized_recurring_price": "$0.00",
"recurring_price": "$0.00",
"item_name": "Belkin iPad Mini Case",
"cart_item_id": "cb6e485fc3211200d68d3b0ac3d3ae35",
"delivery_time": "2 Days"
},
"show_subtotal_price": "true",
"subtotal_title": "Subtotal"
},
"none": {
"subtotal_price": "--",
"subtotal_recurring_frequency": "",
"subtotal_recurring_price": "$0.00",
"total_title": "Total",
"items": [
{
"catalog_item_id": "186d917a6fab7980575967ddeb3ee4f2",
"variables": {
  "Preferred Email address": "abel.tuter@example.com"
},
"quantity": "1",
"localized_price": "$0.00",
"price": "$0.00",
"recurring_frequency": "",
"localized_recurring_price": "$0.00",
"recurring_price": "$0.00",
"item_name": "New Email Account",
"cart_item_id": "17c5241fc3211200d68d3b0ac3d3ae7f",
"delivery_time": "global.Workflow"
},
{
"catalog_item_id": "8b3ae7fedc1be1004ec5c08239e522b",
"variables": {},
"quantity": "1",
"localized_price": "$0.00",
"price": "$0.00",
"recurring_frequency": "",
"localized_recurring_price": "$0.00",
"recurring_price": "$0.00",
"item_name": "Corp VPN",
"cart_item_id": "97c5241fc3211200d68d3b0ac3d3ae7f",
"delivery_time": "2 Days"
},
{
"catalog_item_id": "e1be6dcb4f7b4200086eeed18110c74c",
"variables": {},
"quantity": "1",
"localized_price": "$0.00",
"price": "$0.00",
"recurring_frequency": "",
"localized_recurring_price": "$0.00",
"recurring_price": "$0.00",
"item_name": "External Monitor",
"cart_item_id": "d3c5241fc3211200d68d3b0ac3d3ae7f",
"delivery_time": "2 Days"
},
{
"catalog_item_id": "962967674ff38200086eeed18110c7e7",
"variables": {},
"quantity": "1",
"localized_price": "$0.00",
"price": "$0.00",
"recurring_frequency": "",
"localized_recurring_price": "$0.00",
"recurring_price": "$0.00",
"item_name": "anything",
"cart_item_id": "962967674ff38200086eeed18110c7e7",
"delivery_time": "2 Days"
}];
"variables": {
  "Please describe any ergonomic requirements\n  Office chair should have lumbar support"
},
"quantity": "1",
"localized_price": "$0.00",
"price": "$0.00",
"recurring_frequency": "",
"localized_recurring_price": "$0.00",
"recurring_price": "$0.00",
"item_name": "Desk Set Up",
"cart_item_id": "d7c5241fc3211200d68d3b0ac3d3ae7f",
"delivery_time": "5 Days"
},
"show_subtotal_price": "false",
"subtotal_title": "Subtotal"
},
"yearly": {
  "subtotal_price": "$3,300.00",
  "subtotal_recurring_frequency": "Annually",
  "subtotal_recurring_price": "$300.00",
  "total_title": "Total",
  "items": [
    {
      "catalog_item_id": "04b7e94b4f7b4200086eed18110c7fd",
      "variables": {
        "Adobe Photoshop": "",
        "Adobe Acrobat": "true",
        "Optional Software": "",
        "Additional software requirements": "MS Office 2007"
      },
      "quantity": "3",
      "localized_price": "$1,100.00",
      "price": "$1,100.00",
      "recurring_frequency": "Annually",
      "localized_recurring_price": "$100.00",
      "recurring_price": "$100.00",
      "item_name": "Standard Laptop",
      "cart_item_id": "9fc5241fc3211200d68d3b0ac3d3ae7e",
      "delivery_time": "5 Days"
    }
  ],
  "show_subtotal_price": "true",
  "subtotal_title": "Subtotal"
},
"show_subtotal_price": "true",
"subtotal_title": "Subtotal"
}

Case-2: if two-step is true:

{ "result": {
  "request_number": "REQ0010001",
  "request_id": "82a7e89fc3211200d68d3b0ac3d3ae0a"
} }
Sample Python request

```python
# Need to install requests package for python
easy_install requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/v1/servicecatalog/items/6690750f4f7b4200086eed18110c761/checkout_guide'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {"Content-Type":"application/json","Accept":"application/json"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd),
headers=headers ,data='{"items":[
  {"sys_id":"04b7e94b4f7b4200086eed18110c7fd",
   "variables":{
     "acrobat":"true",
     "Additional_software_requirements":"MS Office 2007"
   },
   "sysparm_quantity":"3"
  },
  {"sys_id":"e1be6dcb4f7b4200086eed18110c74c"
  },
  {"sys_id":"186d917a6fab7980575967dadb3ee4f2",
   "variables":{
     "new_email":"abel.tuter@example.com"
   }
  },
  {"sys_id":"8b3ae7fedc1be1004ec5c08239e522b"
  },
  {"sys_id":"962967674ff38200086eed18110c7e7",
   "variables":{
     "ergonomic_office":"Office chair should have lumbar support"
   }
  }
]}
')

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

Case-1: if two-step is false:
```
"result": {
  "cart_id": "0d8c7a32c3211200d68d3b0ac3d3aeba",
  "subtotal_price": "$3,350.00",
  "subtotal_recurring_frequency": "",
  "subtotal_recurring_price": "$300.00",
  "total_title": "Total",
  "monthly": {
    "subtotal_price": "$50.00",
    "subtotal_recurring_frequency": "Monthly",
    "subtotal_recurring_price": "$0.00",
    "total_title": "Total",
    "items": [
      {
        "catalog_item_id": "90af095bcd38798071a208d710d1b64f",
        "variables": {},
        "quantity": "1",
        "localized_price": "$50.00",
        "price": "$50.00",
        "recurring_frequency": "Monthly",
        "localized_recurring_price": "$0.00",
        "recurring_price": "$0.00",
        "item_name": "Belkin iPad Mini Case",
        "cart_item_id": "cb6e485fc3211200d68d3b0ac3d3ae35",
        "delivery_time": "2 Days"
      }
    ],
    "show_subtotal_price": "true",
    "subtotal_title": "Subtotal"
  },
  "none": {
    "subtotal_price": "-",
    "subtotal_recurring_frequency": "",
    "subtotal_recurring_price": "$0.00",
    "total_title": "Total",
    "items": [
      {
        "catalog_item_id": "186d917a6fab7980575967dadb3ee4f2",
        "variables": {
          "Preferred Email address": "abel.tuter@example.com"
        },
        "quantity": "1",
        "localized_price": "$0.00",
        "price": "$0.00",
        "recurring_frequency": "",
        "localized_recurring_price": "$0.00",
        "recurring_price": "$0.00",
        "item_name": "New Email Account",
        "cart_item_id": "17c5241fc3211200d68d3b0ac3d3ae7f",
        "delivery_time": "global.Workflow"
      },
      {
        "catalog_item_id": "8b3ae7fe8e01004ee5c08239e522b",
        "variables": {},
        "quantity": "1",
        "localized_price": "$0.00",
        "price": "$0.00",
        "recurring_frequency": "",
        "localized_recurring_price": "$0.00",
        "recurring_price": "$0.00",
        "item_name": "Corp VPN",
        "cart_item_id": "97c5241fc3211200d68d3b0ac3d3ae7f",
        "delivery_time": "2 Days"
      }
    ]
  }
},
© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
"catalog_item_id": "e1be6dc847b4200086eeed18110c74c",
"variables": {},
"quantity": "1",
"localized_price": "$0.00",
"price": "$0.00",
"recurring_frequency": "",
"localized_recurring_price": "$0.00",
"recurring_price": "$0.00",
"item_name": "External Monitor",
"cart_item_id": "d3c5241fc3211200d68d3b0ac3d3ae7f",
"delivery_time": "2 Days"
},
{}
"catalog_item_id": "962967674ff38200086eeed18110c7e7",
"variables": {
  "Please describe any ergonomic requirements\n\t\t": "Office chair should have lumbar support"
},
"quantity": "1",
"localized_price": "$0.00",
"price": "$0.00",
"recurring_frequency": "",
"localized_recurring_price": "$0.00",
"recurring_price": "$0.00",
"item_name": "Desk Set Up",
"cart_item_id": "d7c5241fc3211200d68d3b0ac3d3ae7f",
"delivery_time": "5 Days"
}],
"show_subtotal_price": "false",
"subtotal_title": "Subtotal"
}
"yearly": {
"subtotal_price": "$3,300.00",
"subtotal_recurring_frequency": "Annually",
"subtotal_recurring_price": "$300.00",
"total_title": "Total",
"items": [
{
"catalog_item_id": "04b7e94b47b4200086eeed18110c7fd",
"variables": {
  "Adobe Photoshop": "",
  "Adobe Acrobat": "true",
  "Optional Software": "",
  "Additional software requirements": "MS Office 2007"
},
"quantity": "3",
"localized_price": "$1,100.00",
"price": "$1,100.00",
"recurring_frequency": "Annually",
"localized_recurring_price": "$100.00",
"recurring_price": "$100.00",
"item_name": "Standard Laptop",
"cart_item_id": "9fc5241fc3211200d68d3b0ac3d3ae7e",
"delivery_time": "5 Days"
}
]{
"show_subtotal_price": "true",
"subtotal_title": "Subtotal"
}
"show_subtotal_price": "true",
"subtotal_title": "Subtotal"}
Case-2: if two-step is true:
{
    "result": {
        "request_number": "REQ0010001",
        "request_id": "82a7e89fc321120d68d3b0ac3d3ae0a"
    }
}

Service Catalog API - POST /sn_sc/servicecatalog/items/{sys_id}/submit_producer

This method creates a record and returns the Table API relative path and redirect url to access the created record.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/items/{sys_id}/submit_producer
Default URL: /api/sn_sc/servicecatalog/items/{sys_id}/submit_producer

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_view</td>
<td>Specifies the device that displays the Item:</td>
</tr>
<tr>
<td></td>
<td>· Desktop(desktop)</td>
</tr>
<tr>
<td></td>
<td>· Mobile(mobile)</td>
</tr>
<tr>
<td></td>
<td>· Desktop and Mobile(both)</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates successful request and returns the response object consisting of Table API and url to access the created record.</td>
</tr>
<tr>
<td>404</td>
<td>Requested resource cannot be found.</td>
</tr>
<tr>
<td>400</td>
<td>Request is not processed due to incorrect request message.</td>
</tr>
<tr>
<td>500</td>
<td>Internal error.</td>
</tr>
</tbody>
</table>

Sample cURL request

```bash
curl "https://instance.service-now.com/api/sn_sc/servicecatalog/items/3f1dd0320a0a0b990000a53f7604a2ef9/submit_producer" \
--request POST \
--header "Accept:application/json" \
--header "Content-Type:application/json" \
--data "{'variables':{"urgency":'2','comments':'This is sample incident'}}"
--user 'admin':'admin'

{
"result": {
"record": "api/now/table/incident/1584f89767211200f579794717415a79,
"redirect_url": "incident.do?sys_id=1584f89767211200f579794717415a79&sysparm_view=ess"
}
}
```

Sample Python request

```python
#Need to install requests package for python
#easy_install requests
import requests
# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/servicecatalog/items/3f1dd0320a0a0b990000a53f7604a2ef9/submit_producer'
# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type':"application/xml","Accept":"application/xml"}
```
```python
response = requests.post(url, auth=(user, pwd), headers=headers, data="{'variables': {'urgency':'2','comments':'This is sample incident'}}")
# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)
```

```xml
<?xml version="1.0" encoding="UTF-8"?
<result>
<record>api/now/table/incident/6db6f89767211200f579794717415a7a</record>
<redirect_url>incident.do?sys_id=6db6f89767211200f579794717415a7a&amp;sysparm_view=ess</redirect_url>
</result>
</response>
```

**Service Catalog API - POST /sn_sc/servicecatalog/items/{sys_id}/order_now**

Orders the specified catalog item.

**URL format**

Versioned URL: `/api/sn_sc/{version}/servicecatalog/items/{sys_id}/order_now`

Default URL: `/api/sn_sc/servicecatalog/items/{sys_id}/order_now`

**Supported request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Optional. Version of the endpoint to access. For example, ’v1’ or ’v2’. Only specify this value to use an endpoint version other than the “latest”.</td>
</tr>
<tr>
<td>sys_id</td>
<td>Unique identifier of the catalog item.</td>
</tr>
</tbody>
</table>

**Headers**

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.
Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Data format of the response body. Supported types: application/json or application/xml. Default: application/json</td>
</tr>
<tr>
<td>Content-Type</td>
<td>Data format of the request body. Supported types: application/json or application/xml. Default: application/json</td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request is placed successfully.</td>
</tr>
<tr>
<td>400</td>
<td>Indicates that the quantity value is invalid and the request is not placed.</td>
</tr>
</tbody>
</table>

Request body

The API accepts these JSON or XML elements in the request body.

Elements accepted in the request body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_quantity</td>
<td>Required. Quantity of the item to order.</td>
</tr>
<tr>
<td>variables</td>
<td>JSON object containing the variable name and the value. If there are mandatory variables defined for the item, they must be passed to the endpoint.</td>
</tr>
</tbody>
</table>

Response body

The API returns these JSON or XML elements in the response body.
Elements returned in the response body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cart_id</td>
<td>Cart that is generated when the two-step checkout is enabled.</td>
</tr>
<tr>
<td>number</td>
<td>Number of the generated request.</td>
</tr>
<tr>
<td>parent_id</td>
<td>If available, the sys_id of the parent record from which the request is created.</td>
</tr>
<tr>
<td>parent_table</td>
<td>Name of the parent table.</td>
</tr>
<tr>
<td>sys_id</td>
<td>Unique identifier of the generated request.</td>
</tr>
<tr>
<td>table</td>
<td>Table name of the request.</td>
</tr>
</tbody>
</table>

Sample cURL request

```bash
curl "https://instance.service-now.com/api/sn_sc/servicecatalog/items/d82ea08510247200964f77ffeec6c4ee/order_now" \ 
  --request POST \ 
  --header "Accept:application/json" \ 
  --header "Content-Type:application/json" \ 
  --data "
  {sysparm_quantity: 1, 
   variables: {
     replacement: 'Yes', 
     originalnumber: '1640000',
     data_plan: '500MB'
   }
 }" \ 
  --user 'username':'password
```

```json
{
  "result": {
    "sys_id": "cf56a3fcdb3a2300e890f71fbf9619ac",
    "number": "REQ0010012",
    "request_number": "REQ0010012",
    "request_id": "cf56a3fcdb3a2300e890f71fbf9619ac",
    "table": "sc_request"
  }
}
```

Sample Python request

```python
# Need to install requests package for python
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/servicecatalog/items/d82ea08510247200964f77ffeec6c4ee/order_now'

# Eg. User name="username", Password="password" for this code sample.
user = 'username'
pwd = 'password'
```
# Set proper headers
headers = {"Content-Type": "application/json", "Accept": "application/json"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd), headers=headers, data="{
    sysparm_quantity: 1,
    variables: {
        replacement: 'Yes',
        originalnumber: '1640000',
        data_plan: '500MB'
    }
}")

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

{
    "result": {
        "sys_id": "cf56a3fcdb3a2300e890f71fbf9619ac",
        "number": "REQ0010012",
        "request_number": "REQ0010012",
        "request_id": "cf56a3fcdb3a2300e890f71fbf9619ac",
        "table": "sc_request"
    }
}

Service Catalog API - PUT /sn_sc/servicecatalog/items/{sys_id}/submit_guide

This method retrieves a list of items based on the needs described for an order guide.

URL format

Versioned URL: /api/sn_sc/v1/servicecatalog/items/{sys_id}/submit_guide
Default URL: /api/sn_sc/servicecatalog/items/{sys_id}/submit_guide

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

### Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

### Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

### Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request has completed successfully and returns a list of items depending on the needs described. If the options result in no items, then an empty array is returned.</td>
</tr>
</tbody>
</table>
| 400         | Indicates that the request is invalid. Could be due to one of the following reasons:  
  - User does not have access to the item.  
  - Mandatory variables of one or more items have not been answered. (Only variables defined as mandatory are honored.) |
| 500         | Internal error while calculating items depending on options chosen. |

Request body

The API requires these JSON or XML elements in the request body to specify a submit guide.
### Elements required in the request body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>Array of variables requested for an item. Each variable is a JSON object consisting of variable name and value pairs.</td>
</tr>
</tbody>
</table>

### Response body

The API returns these JSON or XML elements in the response body to describe the matching catalog items.

### Elements returned in the response body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the catalog item.</td>
</tr>
<tr>
<td>sys_id</td>
<td>SysID of the catalog item.</td>
</tr>
<tr>
<td>type</td>
<td>Item type can be:</td>
</tr>
<tr>
<td></td>
<td>· catalog item</td>
</tr>
<tr>
<td></td>
<td>· record producer item</td>
</tr>
<tr>
<td></td>
<td>· order guide</td>
</tr>
<tr>
<td>short_description</td>
<td>Text on the service catalog homepage or search results page, or the title on the order form of the catalog item.</td>
</tr>
<tr>
<td>description</td>
<td>Full description of catalog item.</td>
</tr>
<tr>
<td>icon</td>
<td>Path of the image that appears as an icon beside the catalog item.</td>
</tr>
<tr>
<td>category</td>
<td>List of categories that the item belongs and each category title and sys_id of the category.</td>
</tr>
<tr>
<td>show_price</td>
<td>A Boolean value, if true, returns the following details.</td>
</tr>
<tr>
<td></td>
<td>· price</td>
</tr>
<tr>
<td></td>
<td>· localized_price</td>
</tr>
<tr>
<td></td>
<td>· recurring_price</td>
</tr>
<tr>
<td></td>
<td>· localized_recurring_price</td>
</tr>
<tr>
<td></td>
<td>· recurring_frequency</td>
</tr>
<tr>
<td></td>
<td>· local_currency</td>
</tr>
<tr>
<td></td>
<td>· price_currency</td>
</tr>
<tr>
<td></td>
<td>· recurring_price_currency</td>
</tr>
<tr>
<td>catalogs</td>
<td>List of all the catalogs the item belongs and each catalog title and sys_id of the catalog.</td>
</tr>
<tr>
<td>variables</td>
<td>List of all the variables of the catalog item.</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ui_policy</td>
<td>List of all Catalog UI policies defined on the catalog item.</td>
</tr>
<tr>
<td>client_script</td>
<td>List of all the catalog client scripts defined on the catalog item.</td>
</tr>
<tr>
<td>data_lookup</td>
<td>Array of data lookups defined on the catalog item.</td>
</tr>
</tbody>
</table>

**Sample cURL request**

```bash
curl "https://instance.service-now.com/api/sn_sc/v1/servicecatalog/items/6690750f4f7b4200086eeed18110c761/submit_guide" \
--request PUT \
--header "Accept:application/json" \
--header "Content-Type:application/json" \
--data "{"variables":{"hiring_manager":"6816f79cc0a8016401c5a33be04be441","hiring_group":"221f3db5c61122840ff4bedd3039cc9","remote":"No","standard_package":"No","laptop_type":"developer"}}" \
--user 'admin':'admin'
```

```json
{
    "result": [
        {
            "short_description": "Lenovo - Carbon x1",
            "icon": "1195c2084f889200086eeed18110c74a.iix",
            "description": "<p class="p1"><font size="3" color="#808080"><span class="s1">x1 Carbon</span></font></p>
            <p class="p1"><font size="2" color="#808080" face="arial,helvetica,sans-serif"><span class="s1">The x1 Carbon is Lenovo’s lightest ThinkPad yet. It provides a QHD display that fights glare and weighs less than three pounds. Ideal for most computing tasks, and highly mobile.</span></font></p>
            <p class="p2"><font size="2" color="#808080" face="arial,helvetica,sans-serif"><span class="s1">Technical Specs:</span></font></p>
            <ul class="ul1"><li class="li3"><font size="2" color="#808080"><span class="s1">Intel core i5 processor</span></font></li><li class="li3"><font size="2" color="#808080"><span class="s1">512GB solid state drive (SSD)</span></font></li><li class="li3"><font size="2" color="#808080"><span class="s1">Backlit keyboard</span></font></li></ul>
            "show_price": true,
            "recurring_price": "$100.00",
            "type": "catalog_item",
            "local_currency": "USD",
            "sys_id": "04b7e94b4ff7b4200086eeed18110c7fd",
            "recurring_price_currency": "USD",
            "localized_price": "$1,100.00",
            "price": "$1,100.00",
            "catalogs": [
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
"sys_id": "e0d08b13c3330100c8b837659bba8fb4",
"title": "Service Catalog",
"recurring_frequency": "Annually",
"name": "Standard Laptop",
"localized_recurring_price": "$100.00",
"category": {
  "sys_id": "d258b953c611227a0146101fb1be7c31",
  "title": "Hardware"
},
"price_currency": "USD",
"show_quantity": false,
"quantity": "1",
"order": "100",
"variables": [
  {
    "label": "Optional Software",
    "type": 0,
    "mandatory": false,
    "displayvalue": "",
    "friendly_type": "container_start",
    "render_label": true,
    "read_only": false,
    "children": [
      {
        "label": "Adobe Acrobat",
        "type": 7,
        "mandatory": false,
        "displayvalue": "false",
        "friendly_type": "check_box",
        "display_type": "CheckBox",
        "render_label": true,
        "read_only": false,
        "pricing_implications": false,
        "name": "acrobat",
        "attributes": "edge_encryption_enabled=true",
        "id": "90b72d4b4f7b4200086eeed18110c701",
        "value": false,
        "help_text": "",
        "max_length": 0
      },
      {
        "label": "Adobe Photoshop",
        "type": 7,
        "mandatory": false,
        "displayvalue": "false",
        "friendly_type": "check_box",
        "display_type": "CheckBox",
        "render_label": true,
        "read_only": false,
        "pricing_implications": false,
        "name": "photoshop",
        "attributes": "edge_encryption_enabled=true",
        "id": "a8b72d4b4f7b4200086eeed18110c701",
        "value": false,
        "help_text": "",
        "max_length": 0
      }
    ]
  },
  {
    "name": "optional_label",
    "attributes": "edge_encryption_enabled=true",
    "id": "10b72d4b4f7b4200086eeed18110c700",
    "value": "",
    "friendly_type": "container_start",
    "render_label": true,
    "read_only": false,
    "children": []
  }
]
}
The striking design of the LG Cinema Screen incorporates an ultra-narrow bezel, so virtually all you see is picture. The viewing experience is more immersive for your favorite movies and games. Make the most of your time, with an LG Full HD LED monitor. <br/><br/>**Item Specs:**<br/>- Screen Size: 27"
- Resolution: 1920 x 1080
- Aspect Ratio: 16:9
- Brightness: 250 cd/m2
- Supports Split Screen

**Specs:**
- **Screen Size:** 27"
- **Resolution:** 1920 x 1080
- **Aspect Ratio:** 16:9
- **Brightness:** 250 cd/m2
- **Supports Split Screen**
"variables": [],
"ui_policy": [],
"client_script": {
  "onChange": [],
  "onSubmit": [],
  "onLoad": []
},
"data_lookup": []
},
{
  "sys_id": "186d917a6fab7980575967dadb3ee4f2",
  "short_description": "New Email Creation",
  "catalogs": [
    {
      "sys_id": "e0d08b13c333010c8b837659bba8fb4",
      "title": "Service Catalog"
    }
  ],
  "name": "New Email Account",
  "icon": "b630e97e6fab7980575967dadb3ee4d2.iix",
  "description": "<p style="font-size: 13px;"><font size="4"><span style="color: #485563;">Service Overview:</span></font></p>
  <p style="font-size: 13px;">Creation of a new email account within our Microsoft Exchange environment. Creation of your email address will also create your Active Directory account. </p>",
  "show_price": false,
  "type": "catalog_item",
  "category": {
    "sys_id": "109f0438c6112276003ae8ac13e7009d",
    "title": "Services"
  },
  "show_quantity": false,
  "quantity": "1",
  "order": "300",
  "variables": [
    {
      "label": "Preferred Email address",
      "type": 6,
      "mandatory": true,
      "displayvalue": "",
      "friendly_type": "single_line_text",
      "display_type": "Single Line Text",
      "render_label": true,
      "read_only": false,
      "name": "new_email",
      "attributes": "edge_encryption_enabled=true",
      "id": "65865e474fbb4200086eeed18110c7dd",
      "value": "",
      "help_text": "",
      "max_length": 0
    }
  ],
  "ui_policy": [],
  "client_script": {
    "onChange": [],
    "onSubmit": [],
    "onLoad": []
  },
  "data_lookup": []
},
{
  "sys_id": "8b3ae7fedc1be1004ece5c08239e522b",
  "name": "New Email Account",
  "icon": "b630e97e6fab7980575967dadb3ee4d2.iix",
  "description": "<p style="font-size: 13px;"><font size="4"><span style="color: #485563;">Service Overview:</span></font></p>
  <p style="font-size: 13px;"><font size="3">Creation of a new email account within our Microsoft Exchange environment. Creation of your email address will also create your Active Directory account. </font></p>"
Remote access to Internal Corporate Systems

Corp VPN

Corp VPN provides VPN (Virtual Private Network) access to all Internal Corporate Systems and to IP Restricted Customer Instances. By utilizing Corp VPN, users will be provided a similar experience to being within one of the Corporate office locations, while offsite.

Please reference the CORP VPN Installation and Support Documentation for instructions on installation, usage and FAQ’s. For all other questions, please contact IT Support.

Desk Set Up for New Hires or Employee Moves

The facilities organization will set up your new desk with the following configuration: Docking Station.

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/sn_sc/v1/servicecatalog/items/6690750f4f7b4200086eeed18110c761/submit_guide'

# Eg. User name="admin", Password="admin" for this code sample.
user = 'admin'
pwd = 'admin'

# Set proper headers
headers = {'Content-Type': 'application/json', 'Accept': 'application/json'}
```
Table API
The Table API allows you to perform create, read, update, and delete (CRUD) operations on existing tables.

Table API - DELETE /now/table/{tableName}/{sys_id}
Deletes the specified record from the specified table.

URL format
Versioned URL: /api/now/{version}/table/{tableName}/{sys_id}
Default URL: /api/now/table/{tableName}/{sys_id}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Optional. Version of the endpoint to access. For example, 'v1' or 'v2'. Only specify this value to use an endpoint version other than the &quot;latest&quot;.</td>
</tr>
<tr>
<td>tableName</td>
<td>Name of the table from which to delete the record.</td>
</tr>
<tr>
<td>sys_id</td>
<td>Unique identifier of the record to delete.</td>
</tr>
</tbody>
</table>
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Data format of the response body. Supported types: application/json or application/xml.</td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

Sample cURL request

curl "https://instance.service-now.com/api/now/table/incident/d977b66a4f411200adf9f8e18110c7b2" \  --request DELETE \  --header "Accept:application/json" \  --user 'username':'password'  

Sample Python request

```python
# Need to install requests package for python
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/table/incident/d977b66a4f411200adf9f8e18110c7b2'

# Eg. User name="username", Password="password" for this code sample.
user = 'username'
```
pwd = 'password'

# Set proper headers
headers = {"Accept":"application/json"}

# Do the HTTP request
response = requests.delete(url, auth=(user, pwd),
                           headers=headers)

# Check for HTTP codes other than 204
if response.status_code != 204:
    print('Status:', response.status_code, 'Headers:',
          response.headers, 'Error Response:',
          response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

None

Table API - GET /now/table/{tableName}

Retrieves multiple records for the specified table.

For basic instructions, see Retrieve existing incidents

URL format

Versioned URL: /api/now/{version}/table/{tableName}

Default URL: /api/now/table/{tableName}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Optional. Version of the endpoint to access. For example, &quot;v1&quot; or &quot;v2&quot;. Only specify this value to use an endpoint version other than the &quot;latest&quot;.</td>
</tr>
<tr>
<td>tableName</td>
<td>Name of the table from which to retrieve the records.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>name-value pairs</td>
<td>Name-value pairs to use to filter the result set. This parameter is mutually exclusive with <code>sysparm_query</code>. For example, instead of using <code>&amp;sysparm_query=active=true</code>, you can simplify the calling statement by using <code>&amp;active=true</code>. You can also use the display value when the field is a choice or reference type field, such as <code>&amp;state=closed</code> instead of <code>&amp;state=7</code>. To specify multiple key-value pairs, separate each with an ampersand, such as <code>&amp;active=true&amp;assigned_to=john.smith</code>.</td>
</tr>
<tr>
<td>sysparm_display_value</td>
<td>Data retrieval operation for reference and choice fields. Based on this value, retrieves the display value and/or the actual value from the database.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>• true: returns display values for all fields.</td>
</tr>
<tr>
<td></td>
<td>• false: returns actual values from the database.</td>
</tr>
<tr>
<td></td>
<td>• all: returns both actual and display values.</td>
</tr>
<tr>
<td></td>
<td>Default: false</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> There is no preferred method for setting this parameter. However, specifying the display value may cause performance issues since it is not reading directly from the database and may include referencing other fields and records. For more information on display values and actual values, see Table API FAQs (KB0534905).</td>
</tr>
<tr>
<td>sysparm_exclude_reference_link</td>
<td>Flag that indicates whether to exclude Table API links for reference fields.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>• true: exclude Table API links for reference fields.</td>
</tr>
<tr>
<td></td>
<td>• false: include Table API links for reference fields.</td>
</tr>
<tr>
<td></td>
<td>Default: false</td>
</tr>
<tr>
<td>sysparm_fields</td>
<td>Comma-separated list of field names to return in the response.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sysparm_limit</td>
<td>Maximum number of records to return. Unusually large <code>sysparm_limit</code> values can impact system performance. For requests that exceed this number of records, use the <code>sysparm_offset</code> parameter to paginate record retrieval. Default 10000.</td>
</tr>
</tbody>
</table>
| sysparm_offset    | Starting record index for which to begin retrieving records. Use this value to paginate record retrieval. This functionality enables the retrieval of all records, regardless of the number of records, in small manageable chunks. For example, the first time this endpoint is called, `sysparm_offset` is set to '0'. To simply page through all available records, use sysparm_offset=sysparm_offset+sysparm_limit, until the end of the all records is reached. You can use table references, such as first, next, and last, to define the relative starting point of the record index. For example:  
  * sysparm_offset=0>;rel="first"  
  * sysparm_offset=2>;rel="next"  
  * sysparm_offset=6>;rel="last"  
<p>|                   | Do not pass a negative number in the <code>sysparm_offset</code> parameter.                                                                                                                                              |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_query</td>
<td>Encoded query used to filter the result set. For example: {(sysparm_query=caller_id=javascript:gs.getUserID()^active=true)}. The encoded query supports <code>order</code> by. To sort responses based on certain fields, use the <code>ORDERBY</code> and <code>ORDERBYDESC</code> clauses in <code>sysparm_query</code>. For example, <code>sysparm_query=active=true^ORDERBYnumber^ORDERBYDESCcategory</code> filters all active records and orders the results in ascending order by number first, and then in descending order by category. If part of the query is invalid, such as by specifying an invalid field name, the instance ignores the invalid part. It then returns rows using only the valid portion of the query. You can control this behavior using the property <code>glide.invalid_query.returns_no_rows</code>. Set this property to true to return no rows on an invalid query.</td>
</tr>
</tbody>
</table>
| sysparm_suppress_pagination_header | Flag that indicates whether to remove the Link header from the response. The Link header provides various URLs to relative pages in the record set which you can use to paginate the returned record set. Valid values:  
  - true: remove the Link header from the response.  
  - false: do not remove the Link header from the response.  
  
  Default: false |

**Note:** This property controls the behavior of all queries across the instance, such as in lists, scripts (`GlideRecord.query()`), and web service APIs.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_view</td>
<td>UI view for which the data is rendered; determines the fields returned in</td>
</tr>
<tr>
<td></td>
<td>the response. Valid values:</td>
</tr>
<tr>
<td></td>
<td>· desktop</td>
</tr>
<tr>
<td></td>
<td>· mobile</td>
</tr>
<tr>
<td></td>
<td>· both</td>
</tr>
</tbody>
</table>

**Note:** If both `sysparm_fields` and `sysparm_view` are specified, the `sysparm_fields` parameter takes priority.

### Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

#### Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Data format of the response body. Supported types: <code>application/json</code> or</td>
</tr>
<tr>
<td></td>
<td><code>application/xml</code>.</td>
</tr>
</tbody>
</table>

#### Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>REST response data can be split into multiple result sets. Where applicable,</td>
</tr>
<tr>
<td></td>
<td>the response header contains different links for the first set, previous</td>
</tr>
<tr>
<td></td>
<td>set, next set, and the last set of records.</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td>https://&lt;instance name&gt;.service-now.com/api/now/table/cmdb_ci?</td>
</tr>
<tr>
<td></td>
<td>sysparm_offset=40&amp;sysparm_limit=10000&gt;;rel=&quot;next&quot;</td>
</tr>
<tr>
<td></td>
<td>https://&lt;instance name&gt;.service-now.com/api/now/table/cmdb_ci?</td>
</tr>
<tr>
<td></td>
<td>sysparm_offset=40&amp;sysparm_limit=10000&gt;;rel=&quot;prev&quot;</td>
</tr>
<tr>
<td></td>
<td>https://&lt;instance name&gt;.service-now.com/api/now/table/cmdb_ci?</td>
</tr>
<tr>
<td></td>
<td>sysparm_offset=0&amp;sysparm_limit=10000&gt;;rel=&quot;first&quot;</td>
</tr>
<tr>
<td></td>
<td>https://&lt;instance name&gt;.service-now.com/api/now/table/cmdb_ci?</td>
</tr>
<tr>
<td></td>
<td>sysparm_offset=2780&amp;sysparm_limit=10000&gt;;rel=&quot;last&quot;</td>
</tr>
<tr>
<td>Header</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>X-Total-Count</td>
<td>Total count of records returned by the query.</td>
</tr>
</tbody>
</table>

### Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see *REST response codes*.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Request completed successfully. If a valid query returned no results, the response body contains only an empty result array.</td>
</tr>
</tbody>
</table>

### Response body

The API returns these JSON or XML elements in the response body. For the list of available return parameters, refer to the table definition (System Definition) of the record you are retrieving.

**Elements returned in the response body**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-value pairs</td>
<td>Field names and values of all parameters within the specified record or those specified in the query parameters.</td>
</tr>
</tbody>
</table>

### Sample cURL request

```bash
curl "https://instance.service-now.com/api/now/table/problem?sysparm_limit=1" \
--request GET \
--header "Accept:application/json" \
--user 'username': 'password'
```

```json
{
"result": [
{
"parent": "",
"made_sla": "true",
"watch_list": "",
"upon_reject": "cancel",
"sys_updated_on": "2016-01-19 04:52:04",
"approval_history": "",
"number": "PRB0000050",
"sys_updated_by": "glide.maint",
"opened_by": {

"link": "https://instance.service-now.com/api/now/table/sys_user/glide.maint",
"value": "glide.maint"
```
Switch occasionally drops connections,
Sample Python request

```
# Need to install requests package for python
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/table/problem?
sysparm_limit=1'

# Eg. User name="username", Password="password" for this code sample.
user = 'username'
pwd = 'password'

# Set proper headers
headers = {'Accept':'application/xml'}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:',response.headers, 'Error Response:', response.content)
    exit()

# Decode the XML response into a dictionary and use the data
print(response.content)
```

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <result>
    <parent />
    <made_sla>false</made_sla>
    <caused_by />
    <watch_list />
    <upon_reject />
    <sys_updated_on>2016-01-19 20:16:07</sys_updated_on>
    <child_incidents />
    <hold_reason />
    <approval_history />
    <number>INC0000001</number>
    <resolved_by>
      <link>https://instance.service-now.com/api/now/table/sys_user/6816f79cc0a8016401c5a33be04be441</link>
      <value>6816f79cc0a8016401c5a33be04be441</value>
    </resolved_by>
    <sys_updated_by>admin</sys_updated_by>
    <opened_by>
      <link>https://instance.service-now.com/api/now/table/sys_user/681ccaf9c0a8016400b98a06818d57c7</link>
      <value>681ccaf9c0a8016400b98a06818d57c7</value>
    </opened_by>
    <user_input />
    <sys_created_on>2014-05-20 18:24:13</sys_created_on>
  </result>
</response>
```
Table API - GET /now/table/{tableName}/{sys_id}
Retrieves the record identified by the specified sys_id from the specified table.

**URL format**

Versioned URL: /api/now/{version}/table/{tableName}/{sys_id}
Default URL: /api/now/table/{tableName}/{sys_id}
## Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Optional. Version of the endpoint to access. For example, ‘v1’ or ‘v2’. Only specify this value to use an endpoint version other than the “latest”.</td>
</tr>
<tr>
<td>tableName</td>
<td>Name of the table from which to retrieve the record.</td>
</tr>
<tr>
<td>sys_id</td>
<td>Unique identifier of the record to retrieve.</td>
</tr>
<tr>
<td>sysparm_display_value</td>
<td>Data retrieval operation for reference and choice fields. Based on this value, retrieves the display value and/or the actual value from the database.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>* true: returns display values for all fields.</td>
</tr>
<tr>
<td></td>
<td>* false: returns actual values from the database.</td>
</tr>
<tr>
<td></td>
<td>* all: returns both actual and display values.</td>
</tr>
<tr>
<td></td>
<td>Default: false</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> There is no preferred method for setting this parameter. However, specifying the display value may cause performance issues since it is not reading directly from the database and may include referencing other fields and records. For more information on display values and actual values, see Table API FAQs (KB0534905).</td>
</tr>
<tr>
<td>sysparm_exclude_reference_link</td>
<td>Flag that indicates whether to exclude Table API links for reference fields.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>* true: exclude Table API links for reference fields.</td>
</tr>
<tr>
<td></td>
<td>* false: include Table API links for reference fields.</td>
</tr>
<tr>
<td></td>
<td>Default: false</td>
</tr>
<tr>
<td>sysparm_fields</td>
<td>Comma-separated list of field names to return in the response.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sysparm_view</td>
<td>UI view for which the data is rendered; determines the fields returned in the response. Valid values: desktop, mobile, both</td>
</tr>
</tbody>
</table>

Note: If both `sysparm_fields` and `sysparm_view` are specified, the `sysparm_fields` parameter takes priority.

## Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

### Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Data format of the response body. Supported types: application/json or application/xml.</td>
</tr>
</tbody>
</table>

### Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

## Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
<tr>
<td>404</td>
<td>Indicates the record is not found or the requesting user does not have access to the record.</td>
</tr>
</tbody>
</table>
Response body

The API returns these JSON or XML elements in the response body. For the list of available return parameters, refer to the table definition (System Definition) of the record that you are retrieving.

Elements returned in the response body

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-value pairs</td>
<td>Field names and values of all parameters within the specified record or those specified in the query parameters.</td>
</tr>
</tbody>
</table>

Sample cURL request

```bash
curl "https://instance.service-now.com/api/now/table/incident/a9e30c7dc61122760116894de7bcc7bd" \
--request GET \
--header "Accept:application/json" \
--user 'username':'password'
```

```json
{
  "result": {
    "upon_approval": "",
    "location": {
      "link": "https://instance.service-now.com/api/now/table/cmnn_location/105cf7f3c611227501e75e08b14a38ba",
      "value": "105cf7f3c611227501e75e08b14a38ba"
    },
    "expected_start": "",
    "reopen_count": "",
    "close_notes": "",
    "additional_assignee_list": "",
    "impact": "1",
    "urgency": "3",
    "correlation_id": "",
    "sys_tags": "",
    "sys_domain": {
      "link": "https://instance.service-now.com/api/now/table/sys_user_group/global",
      "value": "global"
    },
    "description": "",
    "group_list": "",
    "priority": "3",
    "delivery_plan": "",
    "sys_mod_count": "4",
    "work_notes_list": "",
    "business_service": "",
    "follow_up": "",
    "closed_at": "",
    "sla_due": "2015-11-11 22:04:15",
    "delivery_task": "",
    "sys_updated_on": "2015-11-01 22:37:27",
    "parent": "",
    "work_end": "",
    "number": "INC0000046",
    "closed_by": "",
    "work_start": "",
  }
}
```
"calendar_stc": "",
"category": "software",
"business_duration": "",
"incident_state": "1",
"activity_due": "",
"correlation_display": "",
"company": "",
"active": "true",
"due_date": "",
"assignment_group": {
  "link": "https://instance.service-now.com/api/now/table/sys_user_group/8a4dde73c6112278017a6a4baf547aa7",
  "value": "8a4dde73c6112278017a6a4baf547aa7"
},
"caller_id": {
  "link": "https://instance.service-now.com/api/now/table/sys_user/46c6f9efa9fe198101ddf5eed9adf6e7",
  "value": "46c6f9efa9fe198101ddf5eed9adf6e7"
},
"knowledge": "false",
"made_sla": "false",
"comments_and_work_notes": "",
"parent_incident": "",
"state": "1",
"user_input": "",
"sys_created_on": "2015-11-01 22:05:30",
"approval_set": "",
"reassignment_count": "1",
"rfc": "",
"child_incidents": "",
"opened_at": "2015-11-02 22:04:15",
"short_description": "Can't access SFA software",
"order": "",
"sys_updated_by": "glide.maint",
"resolved_by": "",
"notify": "1",
"upon_reject": "",
"approval_history": "",
"problem_id": {
  "link": "https://instance.service-now.com/api/now/table/problem/a9e4890bc6112276003d7a5a5c774a74",
  "value": "a9e4890bc6112276003d7a5a5c774a74"
},
"work_notes": "",
"calendar_duration": "",
"close_code": "",
"sys_id": "a9e30c7dc61122760116894de7bcc7bd",
"approval": "not requested",
"caused_by": "",
"severity": "3",
"sys_created_by": "admin",
"resolved_at": "",
"assigned_to": "",
"business_stc": "",
"wf_activity": "",
"sys_domain_path": "/",
"cmdb_ci": {
  "link": "https://instance.service-now.com/api/now/table/cmdb_ci/a9c0c82d2c6612276018f7705562f9cb0",
  "value": "a9c0c82d2c6612276018f7705562f9cb0"
},
"opened_by": {
  "link": "https://instance.service-now.com/api/now/table/sys_user/46c6f9efa9fe198101ddf5eed9adf6e7"}
"value": "46c6f9efa9fe198101ddffeed9adf6e7",
"subcategory": "",
"rejection_goto": "",
"sys_class_name": "incident",
"watch_list": "",
"time_worked": "",
"contact_type": "phone",
"escalation": "0",
"comments": ""}
}

Sample Python request

# Need to install requests package for python
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/table/incident/a9e30c7dc61122760116894de7bcc7bd'

# Eg. User name="username", Password="password" for this code sample.
user = 'username'
pwd = 'password'

# Set proper headers
headers = {"Accept":"application/xml"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.content)
    exit()

# Decode the XML response into a dictionary and use the data
print(response.content)

<?xml version="1.0" encoding="UTF-8"?>
<response>
  <result>
    <upon_approval />
    <location>
      <link>https://instance.service-now.com/api/now/table/cmn_location/105cf7f3c611227501e75e08b14a38ba</link>
      <value>105cf7f3c611227501e75e08b14a38ba</value>
    </location>
    <expected_start />
    <reopen_count />
    <close_notes />
    <additional_assignee_list />
    <impact>1</impact>
    <urgency>3</urgency>
    <correlation_id />
    <priority>3</priority>
    <sys_tags />
    <sys_domain>

<calendar_duration />
<work_notes />
<close_code />
<approval>not requested</approval>
<sys_id>a9e30c7dc61122760116894de7bcc7bd</sys_id>
<caused_by />
<severity>3</severity>
<sys_created_by>admin</sys_created_by>
<assigned_to />
<resolved_at />
<business_stc />
<cmdb_ci>
  <link>https://instance.service-now.com/api/now/table/cmdb_ci/a9c0c8d2c6112276018f7705562f9cb0</link>
</cmdb_ci>
<sys_domain_path>/</sys_domain_path>
<wf_activity />
<opened_by>
  <link>https://instance.service-now.com/api/now/table/sys_user/46c6f9e9fe198101dd5eed9adf6e7</link>
</opened_by>
<rejection_goto />
<subcategory />
<sys_class_name>incident</sys_class_name>
<watch_list />
<escalation>0</escalation>
<contact_type>phone</contact_type>
<time_worked />
<comments />
</result>
</response>

**Table API - PATCH /now/table/{tableName}/{sys_id}**

Updates the specified record with the name-value pairs identifier in the request body.

**URL format**

Versioned URL: /api/now/{version}/table/{tableName}/{sys_id}
Default URL: /api/now/table/{tableName}/{sys_id}

**Supported request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Optional. Version of the endpoint to access. For example, “v1” or “v2”. Only specify this value to use an endpoint version other than the “latest”.</td>
</tr>
<tr>
<td>tableName</td>
<td>Name of the table in which to update the record.</td>
</tr>
<tr>
<td>sys_id</td>
<td>Unique identifier of the record to update.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_display_value</td>
<td>Data retrieval operation for reference and choice fields. Based on this value, retrieves the display value and/or the actual value from the database.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: false</td>
</tr>
<tr>
<td>sysparm_fields</td>
<td>Comma-separated list of field names to return in the response.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| sysparm_input_display_value   | Flag that indicates whether to set field values using the display value or the actual value. Valid values:  
|                               | • true: treats input values as display values and they are manipulated so they can be stored properly in the database.  
|                               | • false: treats input values as actual values and stored them in the database without manipulation. Default: false |

**Note:**  
• If this parameter is set to true, pay attention to input values, especially date values, as these are interpreted as being supplied via the user time zone preference and are transformed into UTC format.  
• To set the value of an encrypted field, you must set this parameter to true. If this parameter is not set to true, values submitted to encrypted fields are not saved. Additionally, the requesting user must have the appropriate encryption context prior to submitting the request. Encrypted fields are hidden for users without the appropriate encryption context. For more information on display values and actual values, see Table API FAQs (KB0534905). For more information on field encryption see Encryption support.

| sysparm_view                   | UI view for which the data is rendered; determines the fields returned in the response. Valid values:  
|                               | • desktop  
|                               | • mobile  
|                               | • both |

**Note:** If both `sysparm_fields` and `sysparm_view` are specified, the `sysparm_fields` parameter takes priority.
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Data format of the response body. Supported types: application/json or application/xml.</td>
</tr>
<tr>
<td>Content-Type</td>
<td>Data format of the request body. Supported types: application/json or application/xml.</td>
</tr>
<tr>
<td>X-no-response-body</td>
<td>By default, responses include body content detailing the modified record. Set this request header to true to suppress the response body.</td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

Request body

The API accepts these JSON or XML elements in the request body. For the list of available parameters for the specified record, refer to its Tables definition (System Definition).
Elements accepted in the request body

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-value pairs</td>
<td>Field name and the new value for each parameter to update in the specified record. All fields within a record may not be available for update. For example, fields that have a prefix of &quot;sys_&quot; are typically system parameters that are automatically generated and cannot be updated.</td>
</tr>
</tbody>
</table>

Response body

The API returns these JSON or XML elements in the response body. For the list of available return parameters, refer to the table definition (System Definition) of the record for which you are making modifications.

Elements returned in the response body

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-value pairs</td>
<td>Field names and values of all parameters within the specified record or those specified in the query parameters.</td>
</tr>
</tbody>
</table>

Sample cURL request

```bash
curl "https://instance.service-now.com/api/now/table/incident/ef43c6d40a0b5700c77f9bf387afe3" \
--request PATCH \
--header "Accept:application/json" \
--header "Content-Type:application/json" \
--data "{'assigned_to':'681b365ec0a80164000fb05854a0cd','urgency':'1','comments':'Elevating urgency, this is a blocking issue'}" \
--user 'username':'password'
```

```json
{
  "result": {
    "upon_approval": "proceed",
    "location": {
      "link": "https://instance.service-now.com/api/now/table/cmn_location/108752c8c611227501d4ab0e392ba97f",
      "value": "108752c8c611227501d4ab0e392ba97f"
    },
    "expected_start": "",
    "reopen_count": "",
    "close_notes": "",
    "additional_assignee_list": "",
    "impact": "1",
    "urgency": "1",
    "correlation_id": "",
    "sys_tags": "",
    "sys_domain": {
```
Can't access Exchange server - is it down?
<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>problem_id</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>work_notes</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>calendar_duration</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>close_code</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>sys_id</code></td>
<td>&quot;ef43c6d40a0a0b5700c77f9bf387afe3&quot;</td>
</tr>
<tr>
<td><code>approval</code></td>
<td>&quot;not requested&quot;</td>
</tr>
<tr>
<td><code>caused_by</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>severity</code></td>
<td>&quot;3&quot;</td>
</tr>
<tr>
<td><code>sys_created_by</code></td>
<td>&quot;glide.maint&quot;</td>
</tr>
<tr>
<td><code>resolved_at</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>assigned_to</code></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;link&quot;: &quot;<a href="https://instance.service-now.com/api/now/table/sys_user/681b365ec0a80164000fb0b05854a0cd">https://instance.service-now.com/api/now/table/sys_user/681b365ec0a80164000fb0b05854a0cd</a>&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;value&quot;: &quot;681b365ec0a80164000fb0b05854a0cd&quot;</td>
</tr>
<tr>
<td></td>
<td>},</td>
</tr>
<tr>
<td><code>business_stc</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>wf_activity</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>sys_domain_path</code></td>
<td>&quot;/&quot;</td>
</tr>
<tr>
<td><code>cmdb_ci</code></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;link&quot;: &quot;<a href="https://instance.service-now.com/api/now/table/cmdb_ci/281190e3c0a8000b003f593aa3f20ca6">https://instance.service-now.com/api/now/table/cmdb_ci/281190e3c0a8000b003f593aa3f20ca6</a>&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;value&quot;: &quot;281190e3c0a8000b003f593aa3f20ca6&quot;</td>
</tr>
<tr>
<td></td>
<td>},</td>
</tr>
<tr>
<td><code>opened_by</code></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;link&quot;: &quot;<a href="https://instance.service-now.com/api/now/table/sys_user/glide.maint">https://instance.service-now.com/api/now/table/sys_user/glide.maint</a>&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;value&quot;: &quot;glide.maint&quot;</td>
</tr>
<tr>
<td></td>
<td>},</td>
</tr>
<tr>
<td><code>subcategory</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>rejection_goto</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>sys_class_name</code></td>
<td>&quot;incident&quot;</td>
</tr>
<tr>
<td><code>watch_list</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>time_worked</code></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td><code>contact_type</code></td>
<td>&quot;phone&quot;</td>
</tr>
<tr>
<td><code>escalation</code></td>
<td>&quot;3&quot;</td>
</tr>
<tr>
<td><code>comments</code></td>
<td>&quot;&quot;</td>
</tr>
</tbody>
</table>

**Sample Python request**

```python
# Need to install requests package for python
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/table/incident/ef43c6d40a0a0b5700c77f9bf387afe3'

# Eg. User name=“username”, Password=“password” for this code sample.
user = 'username'
pwd = 'password'

# Set proper headers
headers = {"Content-Type":"application/xml","Accept":"application/xml"}

# Do the HTTP request
response = requests.patch(url, auth=(user, pwd), headers=headers, data="<request><entry><assigned_to>681b365ec0a80164000fb0b05854a0cd</entry></request>"
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
assigned_to><urgency>1</urgency><comments>Elevating urgency, this is a blocking issue</comments></entry></request>

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.content)
exit()

# Decode the XML response into a dictionary and use the data
print(response.content)

<?xml version="1.0" encoding="UTF-8"?>
<response>
  <result>
    <upon_approval>proceed</upon_approval>
    <location>
      <link>https://instance.service-now.com/api/now/table/cmn_location/108752c8c611227501d4ab0e392ba97f</link>
      <value>108752c8c611227501d4ab0e392ba97f</value>
    </location>
    <expected_start />
    <reopen_count />
    <close_notes />
    <additional_assignee_list />
    <impact>1</impact>
    <urgency>1</urgency>
    <correlation_id />
    <priority>1</priority>
    <sys_tags />
    <sys_domain>
      <link>https://instance.service-now.com/api/now/table/sys_user_group/global</link>
      <value>global</value>
    </sys_domain>
    <description />
    <group_list />
    <delivery_plan />
    <sys_mod_count>8</sys_mod_count>
    <work_notes_list />
    <follow_up />
    <business_service />
    <closed_at />
    <sla_due>2017-07-05 05:58:24</sla_due>
    <delivery_task />
    <sys_updated_on>2016-01-22 14:14:54</sys_updated_on>
    <parent />
    <work_end />
    <number>INC0000050</number>
    <closed_by />
    <work_start />
    <calendar_stc />
    <category>hardware</category>
    <business_duration />
    <incident_state>2</incident_state>
    <activity_due>2016-01-22 16:14:54</activity_due>
    <correlation_display />
    <company>
      <link>https://instance.service-now.com/api/now/table/core_company/31bea3d53790200044e0bfc8bcbe5dec</link>
      <value>31bea3d53790200044e0bfc8bcbe5dec</value>
    </company>
    <active>true</active>
Can't access Exchange server - is it down?
Table API - POST /now/table/{tableName}

Inserts one record in the specified table. Multiple record insertion is not supported by this method.

URL format

Versioned URL: /api/now/{version}/table/{tableName}
Default URL: /api/now/table/{tableName}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Optional. Version of the endpoint to access. For example, ‘v1’ or ‘v2’. Only specify this value to use an endpoint version other than the “latest”.</td>
</tr>
<tr>
<td>tableName</td>
<td>Name of the table in which to save the record.</td>
</tr>
</tbody>
</table>
| sysparm_display_value    | Data retrieval operation for reference and choice fields. Based on this value, retrieves the display value and/or the actual value from the database. Valid values:  
  - true: returns display values for all fields.
  - false: returns actual values from the database.
  - all: returns both actual and display values. Default: false |

Note: There is no preferred method for setting this parameter. However, specifying the display value may cause performance issues since it is not reading directly from the database and may include referencing other fields and records. For more information on display values and actual values, see Table API FAQs (KB0534905).
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_exclude_reference_link</td>
<td>Flag that indicates whether to exclude Table API links for reference fields.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>- true: exclude Table API links for reference fields.</td>
</tr>
<tr>
<td></td>
<td>- false: include Table API links for reference fields.</td>
</tr>
<tr>
<td></td>
<td>Default: false</td>
</tr>
<tr>
<td>sysparm_fields</td>
<td>Comma-separated list of field names to return in the response.</td>
</tr>
<tr>
<td>sysparm_input_display_value</td>
<td>Flag that indicates whether to set field values using the display value or the actual value.</td>
</tr>
<tr>
<td></td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td>- true: treats input values as display values and they are manipulated so they can be stored properly in the database.</td>
</tr>
<tr>
<td></td>
<td>- false: treats input values as actual values and stored them in the database without manipulation.</td>
</tr>
<tr>
<td></td>
<td>Default: false</td>
</tr>
</tbody>
</table>

**Note:**
- If this parameter is set to `true`, pay attention to input values, especially date values, as these are interpreted as being supplied via the user time zone preference and are transformed into UTC format.
- To set the value of an encrypted field, you must set this parameter to `true`. If this parameter is not set to true, values submitted to encrypted fields are not saved. Additionally, the requesting user must have the appropriate encryption context prior to submitting the request. Encrypted fields are hidden for users without the appropriate encryption context. For more information on display values and actual values, see *Table API FAQs* *(KB0534905)*. For more information on field encryption see *Encryption support*. 

© 2020 ServiceNow, Inc. All rights reserved. 
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_view</td>
<td>UI view for which the data is rendered; determines the fields returned in the response. Valid values: desktop, mobile, both</td>
</tr>
</tbody>
</table>

**Note:** If both `sysparm_fields` and `sysparm_view` are specified, the `sysparm_fields` parameter takes priority.

### Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

#### Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Data format of the response body. application/json or application/xml.</td>
</tr>
<tr>
<td>Content-Type</td>
<td>Data format of the request body. application/json or application/xml.</td>
</tr>
<tr>
<td>X-no-response-body</td>
<td>By default, responses include body content detailing the new record. Set this header to true in the request to suppress the response body.</td>
</tr>
</tbody>
</table>

#### Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>Location of the created resource.</td>
</tr>
</tbody>
</table>

### Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

#### Status codes

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>
Request body

The API accepts these JSON or XML elements in the request body. For the list of available parameters for the specified record, refer to its Tables definition (System Definition).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-value pairs</td>
<td>Field name and the associated value for each parameter to update in the specified record. All fields within a record may not be available for update. For example, fields that have a prefix of &quot;sys_&quot; are typically system parameters that are automatically generated and cannot be updated. Fields that are not specified and not auto generated by the system are set to the associated data type's null value.</td>
</tr>
</tbody>
</table>

Response body

The API returns these JSON or XML elements in the response body. For the list of available return parameters, refer to the table definition (System Definition) of the record for which you are creating.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-value pairs</td>
<td>Field names and values of all parameters within the specified record or those specified in the query parameters.</td>
</tr>
</tbody>
</table>

Sample cURL request

```bash
curl "https://instance.service-now.com/api/now/table/incident" \
--request POST \
--header "Accept:application/json" \
--header "Content-Type:application/json" \
--data "{'short_description':'Unable to connect to office wifi','assignment_group':'287ebd7da9fe198100f92cc8d1d2154e','urgency':'2','impact':'2'}" \
--user 'username':'password'
```

```json
{
    "result": {
        "upon_approval": "proceed",
        "location": "",
        "expected_start": "",
        "reopen_count": "0",
        "close_notes": "",
        "additional_assignee_list": "",
        "impact": "2",
        "urgency": "2"
    }
}
```
"correlation_id": "",
"sys_tags": "",
"sys_domain": {
    "link": "https://instance.service-now.com/api/now/table/sys_user_group/global",
    "value": "global"
},
"description": "",
"group_list": "",
"priority": "3",
"delivery_plan": "",
"sys_mod_count": "0",
"work_notes_list": "",
"business_service": "",
"follow_up": "",
"closed_at": "",
"sla_due": "",
"delivery_task": "",
"sys_updated_on": "2016-01-22 14:28:24",
"parent": "",
"work_end": "",
"number": "INC0010002",
"closed_by": "",
"work_start": "",
"calendar_stc": "",
"category": "inquiry",
"business_duration": "",
"incident_state": "1",
"activity_due": "",
"correlation_display": "",
"company": "",
"active": "true",
"due_date": "",
"assignment_group": {
    "link": "https://instance.service-now.com/api/now/table/sys_user_group/287ebd7da9fe198100f92cc8d1d2154e",
    "value": "287ebd7da9fe198100f92cc8d1d2154e"
},
"caller_id": "",
"knowledge": "false",
"made_sla": "true",
"comments_and_work_notes": "",
"parent_incident": "",
"state": "1",
"user_input": "",
"sys_created_on": "2016-01-22 14:28:24",
"approval_set": "",
"reassignment_count": "0",
"rfc": "",
"child_incidents": "0",
"opened_at": "2016-01-22 14:28:24",
"short_description": "Unable to connect to office wifi",
"order": "",
"sys_updated_by": "admin",
"resolved_by": "",
"notify": "1",
"upon_reject": "cancel",
"approval_history": "",
"problem_id": "",
"work_notes": "",
"calendar_duration": "",
"close_code": "",
"sys_id": "c537bae64f411200adf9f8e18110c76e",
"approval": "not requested"
Sample Python request

```python
# Need to install requests package for python
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/table/incident

# Eg. User name="username", Password="password" for this code sample.
user = 'username'
pwd = 'password'

# Set proper headers
headers = {"Content-Type":"application/xml", "Accept":"application/json"}

# Do the HTTP request
response = requests.post(url, auth=(user, pwd), headers=headers, data="<request><entry><short_description>Unable to connect to office wifi</short_description><assignment_group>287ebd7da9fe198100f92cc8did2154e</assignment_group><urgency>2</urgency><impact>2</impact></entry></request>"

# Check for HTTP codes other than 201
if response.status_code != 201:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
```
print(data)

{
    "result": {
        "upon_approval": "proceed",
        "location": "",
        "expected_start": "",
        "reopen_count": "0",
        "close_notes": "",
        "additional_assignee_list": "",
        "impact": "2",
        "urgency": "2",
        "correlation_id": "",
        "sys_tags": "",
        "sys_domain": {
            "link": "https://instance.service-now.com/api/now/table/sys_user_group/global",
            "value": "global"
        },
        "description": "",
        "group_list": "",
        "priority": "3",
        "delivery_plan": "",
        "sys_mod_count": "0",
        "work_notes_list": "",
        "business_service": "",
        "follow_up": "",
        "closed_at": "",
        "sla_due": "",
        "delivery_task": "",
        "sys_updated_on": "2016-01-22 14:28:24",
        "parent": "",
        "work_end": "",
        "number": "INC0010002",
        "closed_by": "",
        "work_start": "",
        "calendar_stc": "",
        "category": "inquiry",
        "business_duration": "",
        "incident_state": "1",
        "activity_due": "",
        "correlation_display": "",
        "company": "",
        "active": "true",
        "due_date": "",
        "assignment_group": {
            "link": "https://instance.service-now.com/api/now/table/sys_user_group/287ebd7da9fe198100f92cc8d1d2154e",
            "value": "287ebd7da9fe198100f92cc8d1d2154e"
        },
        "caller_id": "",
        "knowledge": "false",
        "made_sla": "true",
        "comments_and_work_notes": "",
        "parent_incident": "",
        "state": "1",
        "user_input": "",
        "sys_created_on": "2016-01-22 14:28:24",
        "approval_set": "",
        "reassignment_count": "0",
        "rfc": "",
        "child_incidents": "0",
        "opened_at": "2016-01-22 14:28:24",
    }
}
"short_description": "Unable to connect to office wifi",
"order": "",
"sys_updated_by": "admin",
"resolved_by": "",
"notify": "1",
"upon_reject": "cancel",
"approval_history": "",
"problem_id": "",
"work_notes": "",
"calendar_duration": "",
"close_code": "",
"sys_id": "c537bae64f411200adf9f8e18110c76e",
"approval": "not requested",
"caused_by": "",
"severity": "3",
"sys_created_by": "admin",
"resolved_at": "",
"assigned_to": "",
"business_stc": "",
"wf_activity": "",
"sys_domain_path": "/",
"cmdb_ci": "",
opened_by": {
  "link": "https://instance.service-now.com/api/now/table/sys_user/6816f79cc0a8016401c5a33be04be441",
  "value": "6816f79cc0a8016401c5a33be04be441"
},
"subcategory": "",
"rejection_goto": "",
"sys_class_name": "incident",
"watch_list": "",
"time_worked": "",
"contact_type": "phone",
"escalation": "0",
"comments": ""
}

Table API - PUT /now/table/(tableName)/{sys_id}

Updates the specified record with the request body.

URL format

Versioned URL: /api/now/{version}/table/{tableName}/{sys_id}
Default URL: /api/now/table/{tableName}/{sys_id}

Supported request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Optional. Version of the endpoint to access. For example, 'v1' or 'v2'. Only specify this value to use an endpoint version other than the &quot;latest&quot;.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tableName</td>
<td>Name of the table in which to update the record.</td>
</tr>
<tr>
<td>sys_id</td>
<td>Unique identifier of the record to update.</td>
</tr>
<tr>
<td>sysparm_display_value</td>
<td>Data retrieval operation for reference and choice fields. Based on this value, retrieves the display value and/or the actual value from the database. Valid values:  · true: returns display values for all fields.  · false: returns actual values from the database.  · all: returns both actual and display values. Default: false</td>
</tr>
<tr>
<td>sysparm_exclude_reference_link</td>
<td>Flag that indicates whether to exclude Table API links for reference fields. Valid values:  · true: exclude Table API links for reference fields.  · false: include Table API links for reference fields. Default: false</td>
</tr>
<tr>
<td>sysparm_fields</td>
<td>Comma-separated list of field names to return in the response.</td>
</tr>
</tbody>
</table>

**Note:** There is no preferred method for setting this parameter. However, specifying the display value may cause performance issues since it is not reading directly from the database and may include referencing other fields and records. For more information on display values and actual values, see Table API FAQs (KB0534905).
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_input_display_value</td>
<td>Flag that indicates whether to set field values using the display value or the actual value. Valid values:</td>
</tr>
<tr>
<td></td>
<td>• true: treats input values as display values and they are manipulated so they can be stored properly in the database.</td>
</tr>
<tr>
<td></td>
<td>• false: treats input values as actual values and stored them in the database without manipulation.</td>
</tr>
<tr>
<td></td>
<td>Default: false</td>
</tr>
</tbody>
</table>

**Note:**
- If this parameter is set to **true**, pay attention to input values, especially date values, as these are interpreted as being supplied via the user time zone preference and are transformed into UTC format.
- To set the value of an encrypted field, you must set this parameter to **true**. If this parameter is not set to true, values submitted to encrypted fields are not saved. Additionally, the requesting user must have the appropriate encryption context prior to submitting the request. Encrypted fields are hidden for users without the appropriate encryption context. For more information on display values and actual values, see [Table API FAQs (KB0534905)](KB0534905). For more information on field encryption see [Encryption support](#).

| sysparm_view                  | UI view for which the data is rendered; determines the fields returned in the response.                                                                                                                      |
|                              | Valid values:                                                                                                                                                                                                 |
|                              | • desktop                                                                                                                                                                                                     |
|                              | • mobile                                                                                                                                                                                                     |
|                              | • both                                                                                                                                                                                                       |

**Note:** If both **sysparm_fields** and **sysparm_view** are specified, the **sysparm_fields** parameter takes priority.
Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Data format of the response body. <strong>application/json</strong> or <strong>application/xml</strong>.</td>
</tr>
<tr>
<td>Content-Type</td>
<td>Data format of the request body. <strong>application/json</strong> or <strong>application/xml</strong>.</td>
</tr>
<tr>
<td>X-no-response-body</td>
<td>By default, responses include body content detailing the modified record. Set this header to true in the request to suppress the response body.</td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates the request completed successfully.</td>
</tr>
</tbody>
</table>

Response body

The API returns these JSON or XML elements in the response body. For the list of available return parameters, refer to the table definition (System Definition) of the record for which you are creating.

Elements returned in the response body

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-value pairs</td>
<td>Field names and values of all parameters within the specified record or those specified in the query parameters.</td>
</tr>
</tbody>
</table>
Sample cURL request

curl "https://instance.service-now.com/api/now/table/incident/ef43c6d40a0a0b5700c77f9bf387afe3" \
  --request PUT \
  --header "Accept:application/json" \
  --header "Content-Type:application/json" \
  --data "{"assigned_to":"681b365ec0a80164000fb0b05854a0cd","urgency":"1","comments":"Elevating urgency, this is a blocking issue"}"
\n--user 'username': 'password'

{
  "result": {
    "upon_approval": "proceed",
    "location": {
      "link": "https://instance.service-now.com/api/now/table/cmn_location/108752c8c611227501d4ab0e392ba97f",
      "value": "108752c8c611227501d4ab0e392ba97f"
    },
    "expected_start": "",
    "reopen_count": "",
    "close_notes": "",
    "additional_assignee_list": "",
    "impact": "1",
    "urgency": "1",
    "correlation_id": "",
    "sys_tags": "",
    "sys_domain": {
      "link": "https://instance.service-now.com/api/now/table/sys_user_group/global",
      "value": "global"
    },
    "description": "",
    "group_list": "",
    "priority": "1",
    "delivery_plan": "",
    "sys_mod_count": "7",
    "work_notes_list": "",
    "business_service": "",
    "follow_up": "",
    "closed_at": "",
    "sla_due": "2017-07-05 05:58:24",
    "delivery_task": "",
    "sys_updated_on": "2016-01-22 14:12:37",
    "parent": "",
    "work_end": "",
    "number": "INC0000050",
    "closed_by": "",
    "work_start": "",
    "calendar_stc": "",
    "category": "hardware",
    "business_duration": "",
    "incident_state": "2",
    "activity_due": "2016-01-22 16:12:37",
    "correlation_display": "",
    "company": {
      "link": "https://instance.service-now.com/api/now/table/core_company/31bea3d53790200044e0bfc8bcbe5dec",
      "value": "31bea3d53790200044e0bfc8bcbe5dec"
    },
    "active": "true",
  }
Can't access Exchange server - is it down?

severity: 3
opened_at: 2015-11-02 21:58:24
short_description: "Can't access Exchange server - is it down?"
Sample Python request

```python
# Need to install requests package for python
# easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/now/table/incident/ef43c64d0a0b5700c77f9bf387afe3'

# Eg. User name="username", Password="password" for this code sample.
user = 'username'
pwd = 'password'

# Set proper headers
headers = {"Content-Type":"application/xml","Accept":"application/xml"}

# Do the HTTP request
response = requests.put(url, auth=(user, pwd), headers=headers,
data="<request><entry><assigned_to>681b365ec0a80164000fb0b05854a0cd</assigned_to><urgency>1</urgency><comments>Elevating urgency, this is a blocking issue</comments></entry></request>")

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:', response.content)
    exit()

# Decode the XML response into a dictionary and use the data
print(response.content)
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <result>
    <upon_approval>proceed</upon_approval>
    <location>
      <link>https://instance.service-now.com/api/now/table/cmn_location/108752c8c611227501d4ab0e392ba97f</link>
    </location>
    <expected_start />
    <reopen_count />
    <close_notes />
    <additional_assignee_list />
    <impact>1</impact>
    <urgency>1</urgency>
    <correlation_id />
    <priority>1</priority>
    <sys_tags />
    <sys_domain>
      <link>https://instance.service-now.com/api/now/table/sys_user_group/global</link>
    </sys_domain>
  </result>
</response>
```
<value>global</value>
</sys_domain>
<description />
<group_list />
<delivery_plan />
/sys_mod_count>8</sys_mod_count>
<work_notes_list />
<follow_up />
<business_service />
<closed_at />
<sla_due>2017-07-05 05:58:24</sla_due>
<delivery_task />
<sys_updated_on>2016-01-22 14:14:54</sys_updated_on>
<parent />
<work_end />
<number>INC0000050</number>
<closed_by />
<work_start />
<calendar_stc />
<category>hardware</category>
<business_duration />
/incident_state>2</incident_state>
<activity_due>2016-01-22 16:14:54</activity_due>
<correlation_display />
<company>
  <link>https://instance.service-now.com/api/now/table/core_company/31bea3d53790200044e0bfc8bcbe5dec</link>
  <value>31bea3d53790200044e0bfc8bcbe5dec</value>
</company>
<active>true</active>
<due_date />
<assignment_group>
  <link>https://instance.service-now.com/api/now/table/sys_user_group/8a5055c9c61122780043563ef53438e3</link>
  <value>8a5055c9c61122780043563ef53438e3</value>
</assignment_group>
<caller_id>
  <link>https://instance.service-now.com/api/now/table/sys_user/5b7c200da640069006b3845b5d0fa7c</link>
  <value>5b7c200da640069006b3845b5d0fa7c</value>
</caller_id>
<knowledge>false</knowledge>
<made_sla>true</made_sla>
<comments_and_work_notes />
<parent_incident />
<state>2</state>
<user_input />
/sys_created_on>2015-11-02 18:05:40</sys_created_on>
<approval_set />
<reassignment_count>0</reassignment_count>
<rfc />
<opened_at>2015-11-02 21:58:24</opened_at>
<child_incidents />
<order />
<short_description>Can't access Exchange server - is it down?</short_description>
/resolved_by />
/sys_updated_by>admin</sys_updated_by>
<notify>1</notify>
<upon_reject>cancel</upon_reject>
<approval_history />
<problem_id />
<calendar_duration />
<work_notes />
User Role Inheritance API

The User Role Inheritance API allows you to see the roles that a specific user inherited.

This API requires the Contextual Security: Role Management V2 REST API (com.glide.role_management.inh_count.rest_api) plugin, automatically activated on new instances starting with the Jakarta release, or the Contextual Security: Role Management Enhancements REST API plugin, automatically activated starting with the Geneva release. The Contextual Security: Role Management V2 REST API is dependent on the Contextual Security: Role Management V2 (com.glide.role_management.inh_count) plugin.

A user’s role can be directly granted, inherited from other roles, or inherited from groups. You must have the user_admin role to access this API.

User Role Inheritance API - GET /global/user_role_inheritance

Returns a specified user’s granted and inherited roles.

URL format

Versioned URL: /api/global/v1/user_role_inheritance
Default URL: /api/global/user_role_inheritance
Supported request parameters

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_sysid</td>
<td>Required. sys_id of the user for which to return role information.</td>
</tr>
</tbody>
</table>

Headers

The following request and response headers apply to this HTTP action only, or apply to this action in a distinct way. For a list of general headers used in the REST API, see Supported REST API headers.

Request headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>application/json or application/xml</td>
</tr>
</tbody>
</table>

Response headers

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Status codes

The following status codes apply to this HTTP action. For a list of possible status codes used in the REST API, see REST response codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request completed successfully.</td>
</tr>
</tbody>
</table>

Response body

The API returns these JSON or XML elements in the response body.

Elements returned in the response body

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_name</td>
<td>Full name of user.</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>from_role</td>
<td>Roles that are either granted directly or inherited from other roles.</td>
</tr>
<tr>
<td>from_group</td>
<td>Roles that are inherited from groups that have roles.</td>
</tr>
</tbody>
</table>

Sample cURL request

```bash
curl "https://instance.service-now.com/api/global/user_role_inheritance?user_sysid=62826bf03710200044e0bfc8bcbe5df1" \
--request GET \
--header "Accept:application/json" \
--user 'username':'password'
```

```json
{
    "result": {
        "user_name": "abel.tuter",
        "from_group": [
            "[Azure Operators]/sn_azure.operator",
            "[Azure Operators]/sn_azure.operator/sn_azure.user"
        ],
        "from_role": [
            "/snc_internal",
            "/cloud_user",
            "/cloud_user/pa_viewer",
            "/cloud_user/sn_azure.user",
            "/cloud_user/aws_user"
        ]
    }
}
```

Sample Python request

```python
#Need to install requests package for python 
#easy_install requests
import requests

# Set the request parameters
url = 'https://instance.service-now.com/api/global/user_role_inheritance?
user_sysid=62826bf03710200044e0bfc8bcbe5df1'

# Eg. User name="username", Password="password" for this code sample.
user = 'username'
pwd = 'password'

# Set proper headers
headers = {"Content-Type":"application/json","Accept":"application/json"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers)

# Check for HTTP codes other than 200
```
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:',
    response.headers, 'Error Response:', response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
data = response.json()
print(data)

{
    "result": {
        "user_name": "abel.tuter",
        "from_group": [
            "[Azure Operators]/sn_azure.operator",
            "[Azure Operators]/sn_azure.operator/sn_azure.user"
        ],
        "from_role": [
            "/snc_internal",
            "/cloud_user",
            "/cloud_user/pa_viewer",
            "/cloud_user/sn_azure.user",
            "/cloud_user/aws_user"
        ]
    }
}

Scripts

Use scripts to extend your instance beyond standard configurations. With scripts, you may automate processes, add functionality, integrate your instance with an outside application and more.

APIs (Application Programming Interfaces) provide classes and methods that you can use in scripts to define functionality. ServiceNow provides APIs as JavaScript classes, web services, and other points of connection for integrations. Note that you cannot access commonly used JavaScript objects (such as DOM or Window). Jelly scripts are also used in some modules. Jelly is used to turn XML into HTML and may include both client-side and server-side scripts.

Scripts may be server-side (run on the server or database), client-side (run in the user’s browser) or on the MID server.

Note: When you are writing scripts, you cannot use reserved words.

It is recommended that you become familiar with JavaScript coding before you begin customizing your instance, and with Jelly if you intend to deploy Jelly scripts. The following topics provide general information, procedures, and contexts for scripting in the platform.

- Glide class overview
- Syntax editor
- Execution order of scripts and engines
- Glide stack
- Scripting Alert, Info, and Error Messages
- Use business rules and client scripts to control field values
- Scripting of field types

Jelly is also available to turn XML into HTML. Use Jelly in scripts.
When developing scripts for scoped applications, you must use the scoped APIs, which include scoped versions of the Glide APIs. The scoped Glide APIs do not provide all the methods included in the global Glide APIs, and you cannot call a global Glide API in a scoped application.

Available script types

Scripts can be used in many places. The most important detail is whether the script runs on the client or the server.

**Script types and where they run**

<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
<th>Runs on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Control</td>
<td>Determines whether access will be granted for a specified operation to a specific entity.</td>
<td>server - script and any condition run on the server</td>
</tr>
<tr>
<td></td>
<td>- type of entity being secured</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- operation being secured</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- unique identifier describing the object</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Can be defined by roles, conditional expressions or scripts.</td>
<td></td>
</tr>
<tr>
<td>Ajax Scripts</td>
<td>Enables the client to get data from the server to dynamically incorporate into a page without reloading the whole page.</td>
<td>client - Ajax Client Scripts run on the client server - Ajax Server Scripts run on the server</td>
</tr>
<tr>
<td></td>
<td>- Ajax Client Scripts request that information be returned, or that action be taken, or sometimes both</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Ajax Server Scripts fulfill Ajax Client Script requests</td>
<td></td>
</tr>
<tr>
<td>Script</td>
<td>Description</td>
<td>Runs on</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td><strong>Business Rules</strong></td>
<td>Customizes system behavior</td>
<td>server - script and any condition run on the server</td>
</tr>
<tr>
<td></td>
<td>• runs when a database action occurs (query, insert, update or delete)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• the script may run</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• before or after the database action is performed (runs as part of the database operation)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• asynchronously (at some point after the database operation)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• on display (when displaying the data in a form)</td>
<td></td>
</tr>
<tr>
<td><strong>Service catalog UI policy</strong></td>
<td>Defines the display of a variable set or a catalog item (from the service catalog).</td>
<td>client - scripts in the &quot;execute if true&quot; field or &quot;execute if false&quot; field run on the client</td>
</tr>
<tr>
<td></td>
<td></td>
<td>server - all conditions run on the server</td>
</tr>
<tr>
<td><strong>Client Scripts</strong></td>
<td>Used for making changes to the appearance of forms, displaying different fields based on values that are entered or other custom display options.</td>
<td>client</td>
</tr>
<tr>
<td></td>
<td>• onLoad means the Client Script runs when the form or page is loaded</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• onChange means the Client Script runs when something specific gets changed AND also when the form or page loads</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• onSubmit means the Client Script runs when the form is submitted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Client Scripts can also be called by other scripts or modules, including UI policies.</td>
<td></td>
</tr>
<tr>
<td>Script</td>
<td>Description</td>
<td>Runs on</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| **Script actions** | Contains scripts which run when an event occurs, for example  
  - approval is cancelled  
  - change is approved  
  - problem is assigned  
  Can have a condition which must be true for the script to run. Commonly used to call a Script Include. | server - script and any condition run on the server |
| **Script Includes** | Contains scripts which can be functions or classes. These scripts run only when called by other scripts (often Business Rules).  
  Any server script which is complicated or reusable should be a Script Include (especially complicated Business Rules). | server |
| **Transform Maps** | Used for importing data.  
  - defines mapping relationships between tables  
  - can use Business Rules, other scripts and/or other options to import that data  
  Do not always include scripts. | server |
| **UI Actions** | Creates the ability to choose a specific action such as clicking a button or a link.  
  UI Actions put these items on forms and lists:  
  - buttons  
  - links  
  - context menu items  
  - list choices  
  - client - when the "Client" box is checked, the script in the script field runs on the client  
  - server - when the "Client" box is unchecked, the script in the script field runs on the server  
  - client - when the "Client" box is checked, the onClick script is available, which can contain any JavaScript but normally calls a function which is specified in the script field  
  - server - all conditions run on the server |
<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
<th>Runs on</th>
</tr>
</thead>
</table>
| UI Context Menus       | Defines which "right-click menu" will pop-up in which area, and the menu choices that will be available | • client - onShow scripts run on the client  
• client - action scripts run on the client  
• server - dynamic action scripts run on the server  
• server - all conditions run on the server |

**Note:** If you use a left-handed mouse configuration, right-click means "click the other button."
<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
<th>Runs on</th>
</tr>
</thead>
</table>
| UI Macros       | Contains modular, reusable components that can contain Jelly and are called by UI pages. They also contain different types of scripts and may be called multiple times on the same page. | - server - the UI Macro itself executes on the Server  
- server - may contain content that runs on the server (Jelly expressions or JavaScript inside Jelly constructs)  
- client - may generate output that runs on the client (embedded JavaScript within <script> tags) |

**Note:** Jelly turns XML into HTML.
<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
<th>Runs on</th>
</tr>
</thead>
</table>
| UI Pages    | Used to create and display pages, forms, dialogs, lists and other UI components. Can be displayed on a standalone basis, or called as a usable component, as part of a larger page. | • server - Jelly XML runs on the server to produce HTML  
• client - HTML may contain embedded JavaScript that runs on the client  
• client - client scripts run on the client  
• server - processing scripts run on the server |

Can contain
- Client Scripts,
- processing scripts (which are server scripts),
- HTML,
- Jelly,
- UI Macros,
- and also can call other scripts.

**Note:** Jelly turns XML into HTML.
<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
<th>Runs on</th>
</tr>
</thead>
</table>
| UI Policies         | Defines the behavior and visibility of fields on a form.                    | • client - scripts in the "execute if true" field or "execute if false" field run on the client  
                            • server - all conditions run on the server                                      |
|                     | • mandatory                                                                  |                                              |
|                     | • visible                                                                    |                                              |
|                     | • read only                                                                  |                                              |
|                     | Use UI Policies rather than client scripts whenever possible.                 |                                              |
|                     | • UI Policies are always attached to one table                                |                                              |
|                     | • UI Policies often have a condition which must be true in order for them to run |                                              |
| UI Properties       | Designates what the instance will look like.                                 | • server - properties set on the server      
                            • client - the results get rendered on the client                                   |
|                     |                                                                           | no scripts                                  |
| UI Scripts          | Contains client scripts stored for re-use. Only used when called from other scripts. Not recommended for use. | client                                      |
| Validation Scripts  | Validates that values are in a specified format.                             | client                                      |
| Workflow editor     | Used to create or change a workflow. Scripts can be run at any point in a workflow, or different scripts can be run at different points. Scripts also can be found inside every workflow activity and can be modified (although do so with extreme caution). | server - script and any conditions run on the server |

**Execution order of scripts and engines**

Scripts, assignment rules, business rules, workflows, escalations, and engines all take effect in relation to a database operation, such as insert or update. In many cases, the order of these events is important.
The order of execution is as follows:

1. **Before** business rules: Scripts configured to execute before the database operation with an order less than 1000.

2. **Before** engines. The following are not executed in any specific order:
   - Approval engine (for task and sys_approval_approver tables)
   - Assignment rules engine (for task tables)
   - Data policy engine
   - Escalation engine
   - Field normalization engine
   - Role engine - keeps role changes in sync with sys_user_has_role table (for sys_user, sys_user_group, sys_user_grmember, and sys_user_role tables)
   - Execution plan engine (for task tables)
   - Update version engine - creates version entry when sys_update_xml entry is written (for sys_update_xml table)
   - Workflow engine (for default workflows)

3. **Before** business rules: Scripts configured to execute before the database operation with an order greater than or equal to 1000.

4. The database operation (insert, update, delete).

5. **After** business rules: Scripts configured to execute after the database operation with an order less than 1000.

6. **After** engines. The following are not executed in any specific order:
   - Label engine
   - Listener engine
   - Table notifications engine
   - Role engine - keeps role changes in sync with sys_user_has_role table (for sys_user, sys_user_group, sys_user_grmember and sys_user_role tables)
   - Text indexing engine
   - Update sync engine
   - Data lookup engine inserts or updates
   - Workflow engine (for deferred workflows)
   - Trigger engine (for all Flow Designer flows)

7. Email notifications. The following are executed based on the weight of the notification record:
   - Notifications sent on an insert, update, or delete
   - Event-based notifications

8. **After** business rules. Scripts configured to execute after the database operation with an order greater than or equal to 1000.

**Use business rules and client scripts to control field values**

Implement both business rules and client scripts for a field to enable users to set record values properly using both forms and lists, and to see immediate changes to the values in forms as edits are made.
The problem with using only a client script or a business rule to control updates to a field is that fields can be changed on either a form or a list. Client scripts and UI policies run on forms only (client-side) and do not apply to list editing. Allowing list editing with client scripts running on fields in a form can result in incorrect data being saved to the record. For systems in which client scripts or UI policies apply to forms, either disable list editing or create appropriate business rules or access control to control the setting of values in the list editor. A side effect of this is that security measures implemented in client scripts are easy to circumvent. The user only needs to edit the field in a list.

Business rules on a form are not dynamic, the user must update the record for the change to be seen. This makes using client scripts the preferred method for controlling field values on forms.

When using both a business rule and client script to control field values, the update behavior is the same across the system. This means that updated values are not different depending on whether a list of form is used to make the change. This means that the same functionality must be implemented twice, once in a client script and once in a business rule or access control.

Example: Use a business rule to create email addresses during user record import

An organization has a client script that sets the email address for a user to first.last@company.com. The administrator likes this because he can see the email address immediately when he enters the user's information. The administrator then performs a bulk import of users from a spreadsheet containing the users' first and last names. His expectation is that each user's email address will be set automatically, as they are when he edits the form. Since the client script runs only on the form (the interface to the record), it has no effect on data imported into the record from outside that interface, and no email addresses are created. To solve this problem, the administrator implements a business rule that runs when the import occurs and creates the email addresses.

Example: Prevent list edit for a field that is not editable in the form

An organization wants to hide the Priority field on an incident form if the assignment group is Development. They create a UI policy on the incident form to do this, but their users can still see and edit the Priority field using the list editor. To rectify this, apply an access control to prevent read access to the Priority field when the assignment group is Development.

Using NULL as a field value

The string NULL has a particular role in scripts and is a reserved word. The reserved word is NULL in all capital letters. A field with the value Null or null, for example, is acceptable. Only use NULL to clear out a particular field.

Any NULL field values obtained from an import set data source are inserted into the staging table as empty field values. You should not use the term NULL as a field value in import set transform maps or anywhere in the First name or Last name fields. Also, do not use NULL in reference fields as the system interprets the value as a string containing the word NULL, not as a reserved word.

Script evaluation of fields by data type

Script fields evaluate data based on the field type of the input.

<table>
<thead>
<tr>
<th>Type</th>
<th>Evaluates to in script</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The string</td>
<td>&quot;dog&quot; &gt; &quot;dog&quot;</td>
</tr>
<tr>
<td>Type</td>
<td>Evaluates to in script</td>
<td>Example</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Decimal</td>
<td>A number with up to two decimal points</td>
<td>12.34 &gt; 12.34</td>
</tr>
<tr>
<td>Integer</td>
<td>A number with zero decimal points</td>
<td>12 &gt; 12</td>
</tr>
<tr>
<td>True / False</td>
<td>true or false</td>
<td>&gt; true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt; false</td>
</tr>
<tr>
<td>Date</td>
<td>A date formatted as yyyy-mm-dd</td>
<td>2008-11-04</td>
</tr>
<tr>
<td>Date-time</td>
<td>A day and time formatted as yyyy-mm-dd hh:mm:ss</td>
<td>2008-11-04 06:46:20</td>
</tr>
<tr>
<td>Duration</td>
<td>A date that is equal to January 1st 1970 00:00:00 + the amount of time of the duration being stored</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: This date corresponds to the system time zone. If a different user time zone has been specified, the date and time value may appear different for that user.</td>
<td></td>
</tr>
<tr>
<td>Choice</td>
<td>Returns the contents of the value field for the sys_choice record associated with that choice. See: Choice List for more information on returning the value associated with a particular item in a choice list.</td>
<td>&gt; ‘2’ (Note that this value is is a string)</td>
</tr>
<tr>
<td>Journal</td>
<td>Returns a string of all entries made to that journal field. See Journal Fields for scripting of journal type fields</td>
<td>The web server is down &gt; The web server is down</td>
</tr>
<tr>
<td>Reference</td>
<td>Returns the sys_id of the record that is referenced</td>
<td>&gt; ‘287ee6fed9e198100ada7950d0b1b73’</td>
</tr>
<tr>
<td>Image</td>
<td>Returns the path to the image</td>
<td>&gt; images/icons/image_name.gif</td>
</tr>
</tbody>
</table>
### Set a duration field value

Examples of JavaScript that can be used to set the value of a duration field.

> **Note:** Negative duration values are not supported.

### Using the GlideDateTime.subtract() method

The `subtract(GlideDateTime start, GlideDateTime end)` method in enables you to set the duration value using a given start date/time and end date/time. An example on how to set the duration for the time a task was opened is:

```javascript
var duration = GlideDateTime.subtract(start, end);
```

If you want to work with the value returned as a number to use in date or duration arithmetic, convert the return to milliseconds:

```javascript
var time = GlideDateTime.subtract(start,end).getNumericValue();
```

If you want to set a duration to the amount of time between some event and the current date/time:

```javascript
<duration_field> = GlideDateTime.subtract(new GlideDateTime(<start_time>.getValue()),gs.nowDateTime());
```

The time values presented to `GlideDateTime.subtract` are expected to be in the user's timezone and in the user's format.

### Setting a default value of a duration field

Setting the default value for a duration field is similar to the method used in the previous topic.

### Setting the duration field value in a client script

This script sets a `duration_field` value in a client script. Replace `duration_field` with the field name from your instance.

```javascript
g_form.setValue('<duration_field>', '11 01:02:03');
```
Calculating and setting a duration using a client script

Here is an example of how to return a value and populate it using a client script.

Create an `onChange` client script that includes the following code. You can modify this script if you need the calculation to happen in an `onLoad` script or some other way.

```javascript
function onChange(control, oldValue, newValue, isLoading){
    var strt = g_form.getValue('<start_field>');
    var end = g_form.getValue('<end_field>');
    var ajax = new GlideAjax('AjaxDurCalc');
    ajax.addParam('sysparm_name','durCalc');
    ajax.addParam('sysparm_strt',strt);
    ajax.addParam('sysparm_end',end);
    ajax.getXMLWait();
    var answer = ajax.getAnswer();
    g_form.setValue('<duration_field>', answer);
}
```

Create a system script include file called `AjaxDurCalc` that handles the request. It may be reused for other functions as well.

```javascript
var AjaxDurCalc = Class.create();
AjaxDurCalc.prototype = Object.extendObject(AbstractAjaxProcessor,{
durCalc:function(){return GlideDuration.subtract(this.getParameter('sysparm_strt'),this.getParameter('sysparm_end'))} });
```

Changing the duration field value

If you manipulate a duration value with addition/subtraction of some amount of time, use the functions that allow you to get and set the numeric value of the duration. A unit of measure for a duration numeric value is milliseconds. The following is an example that adds 11 seconds to the `duration` field in the current record.

```javascript
var timems = current.duration.dateNumericValue();
timems = timems + 11*1000;
current.duration.setDateNumericValue(timems);
```

Formatting the Resolve Time

To format the Resolve Time or the Business Resolve Time fields as durations, which displays them as a duration instead of a large integer, add the following attribute to those fields:

```javascript
format=glide_duration
```

Modify the dictionary entry for the field and add the attribute. If there is an existing attribute, separate multiple attributes with commas.

Setting the maximum unit of measurement

The `max_unit` dictionary attribute defines the maximum unit of time used in a duration. For example, if `max_unit=minutes`, a duration of 3 hours 5 minutes 15 seconds appears as 185
minutes 15 seconds. To set the maximum unit of duration measurement, add the following
dictionary attribute to the duration field:

\[
\text{max_unit=\langle unit\rangle}
\]

**Using function fields to perform database operations**

Create function fields and scripts in the Now Platform to perform common database
transformations and calculations.

Regular fields store a value in the database. A function field does not store data, but rather
displays the results of a database query. Function fields do not have a database column
associated. Instead, function fields generate a value based on simple computations of other
fields and constants. Function fields can be used like any other regular fields in the system: in
forms, lists, query conditions, reports, and so on.

**Use case**

You want to identify all incidents with a probably useless short description of fewer
than 10 characters.

Rather than query all incident records and then individually determine whether
each record meets the criteria, you can instead create a function field or function
script that retrieves only the records that meet your criteria. This is a much more
efficient query.

For a simple computation, it is better to use a function field or function script rather
than store a computed value. You do not use up space, and the value is always
up to date. An additional benefit of using function fields or function scripts is that
the transformation is performed by the database server rather than the application
node.

**Function fields versus calculated fields**

The values of calculated fields are stored in the database. The values of function fields or function
scripts are not stored in the database, but are computed in real time. Function field values are
always up to date.

**Function fields versus business rules**

When business logic is based on a simple transformation of one or more existing fields, function
fields or function scripts let you implement business logic without storing and maintaining the result
of the transformation using existing platform means such as business rules or calculated fields.

**Function fields versus filters, query strings, and Rhino**

Rather than calling Rhino to perform string transformations, you can use a function field or
function script to perform the transformations for you. Function fields and function scripts are more
efficient and result in up to date values that do not have to be stored or maintained.

**Function field limitations**

Because function field calculations are not actually stored in the database, some operations are
not relevant.
• Function fields cannot be directly audited or indexed.

**Note:** To index a function field as if it were a regular field, make sure the individual fields used by the function are indexed. Or for best performance, make sure there is a composite index including all fields that are used in the function.

• Function fields cannot be encrypted, since a function field value is never stored in the database.
• Function fields cannot be converted to regular fields or vice versa.

### Defining platform functions

Platform functions can be defined in two ways:

1. Use the following application programming interfaces (APIs) to build and use functions in a script.

   **Function APIs**

<table>
<thead>
<tr>
<th>APIs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Scoped GlideDBFunctionBuilder</code></td>
<td>Construct the function to perform a SQL operation.</td>
</tr>
<tr>
<td><code>GlideRecord - addFunction(Object function)</code></td>
<td>Apply the function to a GlideRecord.</td>
</tr>
</tbody>
</table>

2. Create a field that holds the function definition, as shown in the following example.
A dictionary entry manages how ServiceNow stores data in tables and fields (columns). For new dictionary entries, select a Table and the field Type of the new column. Also enter a column label, which becomes the field label, and the column name. If necessary, set a Max length for text type fields, make the field Mandatory to save a record, and make the field a Display Value for reference fields so it appears on records that reference this table. More info

Table: Task [task]

Type: Integer

Column label: short_description_length

Column name: u_short_description_length

Max length: 

Function Definition

Function definition: `glidefunction.length(short_description)`

Submit

Related Links

Advanced view
The Dictionary Entry (sys_dictionary) form has a **Function field** check box which specifies that the field runs a function rather than stores a value.

**glidefunction operations**

Function fields can perform the following operations using the `glidefunction:<operation>` syntax.

**Function field operations**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add()</td>
<td>Takes 2 number fields as input, performs the addition, and returns the results as a field value.</td>
</tr>
<tr>
<td>concat()</td>
<td>Takes any number of fields and constants as input, concatenates the input, and returns a single string as a field value.</td>
</tr>
<tr>
<td>datediff()</td>
<td>Takes 2 date/time fields as input, calculates the difference between the dates in days, minutes, and seconds, and returns the results as a duration field value.</td>
</tr>
<tr>
<td>dayofweek()</td>
<td>Takes two arguments: a date field and a constant of either 1 (week starts on Sunday) or 2 (week starts on Monday). Returns the results as an integer value that represents the day of the week.</td>
</tr>
<tr>
<td>divide()</td>
<td>Takes 2 number fields as input, performs the division, and returns the results as a field value.</td>
</tr>
<tr>
<td>length()</td>
<td>Takes a string field such as a text field as input, calculates the field length in characters, and returns the results as a field value.</td>
</tr>
<tr>
<td>multiply()</td>
<td>Takes 2 number fields as input, performs the multiplication, and returns the results as a field value.</td>
</tr>
<tr>
<td>subtract()</td>
<td>Takes 2 number fields as input, performs the subtraction, and returns the results as a field value.</td>
</tr>
</tbody>
</table>

Create a function field to perform database functions

Create a field that displays the results of a database function, such as a mathematical operation, field length computation, or day of the week calculation.

Role required: personalize_dictionary or admin.

1. Navigate to **System Definition > Dictionary**.
2. Click **New**.
3. Select **Function field**.
4. Fill in these fields on the form.

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
### Fields for function fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Table to which the function field is added.</td>
</tr>
<tr>
<td>Type</td>
<td>Species the type of field: string, number, date, and so on.</td>
</tr>
<tr>
<td>Column label</td>
<td>Defines the label for the column.</td>
</tr>
<tr>
<td>Column name</td>
<td>Defines the name of the column. Created automatically based on <code>Column label</code>.</td>
</tr>
<tr>
<td>Max length</td>
<td>Defines the maximum length of the return value.</td>
</tr>
<tr>
<td>Function field</td>
<td>If selected, creates a field that displays the results of a database function, such as a mathematical operation, field length computation, or day of the week calculation. Once the new function record is saved, you cannot clear the check box to make the field a regular field.</td>
</tr>
</tbody>
</table>

Function field

- **Function definition**
  - Defines the function that the field performs.
  - Starts with `glidefunction:`, followed by the operation to be performed (for example, `concat`), followed by function parameters. Constants must be enclosed in single quotes.
  - For example, the following function definition creates a field that shows the short description, followed by a space, followed by the caller name:

  \[ \text{glidefunction:concat(short\_description, ', ', caller\_id.name)} \]

  **Note:** Function parameters support dot walking. See [Selecting fields on related tables using dot-walking](#).

5. **Click Submit.**
   
   You return to the Dictionary Entry (sys_dictionary) table.

6. **Navigate to the table to which you added the function field.**

7. **Add the function field to the list.** See [Configure items on forms or in lists using a slushbucket](#).

Use the function field like any other field. For example, add it to a form, use it in a filter, or use it in the condition builder.

If the function definition is invalid, instead of the expected value you receive the message **Invalid function** if the function field is a string type field, or an empty value for other field types.

### Using regular expressions in scripts

JavaScript regular expressions automatically use an enhanced regex engine, which provides improved performance and supports all behaviors of standard regular expressions as defined by Mozilla JavaScript. The enhanced regex engine supports using Java syntax in regular expressions.
The `SNC.Regex` API is not available for scoped applications. For scoped applications, remove the `SNC.Regex` API and use standard JavaScript regular expressions.

For more information on JavaScript regular expressions, see the Mozilla JavaScript documentation on regular expressions and `RegExp`.

### Using Java syntax in JavaScript regular expressions

The enhanced regex engine includes an additional flag to allow Java syntax to be used in JavaScript regular expressions.

Regular expressions with the additional flag work in all places that expect a regular expression, such as `String.prototype.split` and `String.prototype.replace`. To use Java syntax in a regular expression, use the Java inline flag `j`, for example `/(?ims)ex(am)ple/j`

#### Extended regular expression flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>j</td>
<td>Defines a regular expression that executes using the Java regular expression engine. It can be used to access Java-only features of regular expressions (such as look behind, negative look behind) or to use Java regular expressions without translating them into JavaScript regular expressions. For example: <code>var regex = /ex(arn)ple/j</code></td>
</tr>
</tbody>
</table>

### Convert SNC Regex expressions to enhanced regex expressions

When you upgrade to Eureka Patch 5 or later releases, you should convert scripts that use the `SNC.Regex` API to use regular JavaScript expressions.

1. From the original expression, such as `SNC.Regex("/expr/is")`, create a new regular expression object using the pattern with the slashes stripped.

   ```javascript
   new RegExp('expr');
   ```

2. Move the `SNC.Regex` flags to the start of the expression using Java’s inline flag special construct.

   ```javascript
   new RegExp('(\is)expr');
   ```

3. Add the `j` flag to the `RegExp` to tell the engine to treat the expression as a Java expression.

   ```javascript
   new RegExp('(\is)expr', 'j');
   ```

   **Note:** If you know that the script being converted does not use Java syntax, it is not necessary to use the `j` flag.

4. Add the `g` flag to handle multiple matches or a global replace.

   ```javascript
   new RegExp('(\is)expr', 'jg');
   ```

---

Using `SNC.Regex`

```javascript
var r = new SNC.Regex('/world/');
var str = 'helloworld';
```
var replaced = r.replaceAll(str, 'there');
// replaced == 'hellothere'

Using a JavaScript regular expression

var r = new RegExp('world', 'jg');
var str = 'helloworld';
var replaced = str.replace(r, 'there');
// replaced == 'hellothere'

Syntax editor

The syntax editor provides support for editing JavaScript scripts. The syntax editor has these features.

- JavaScript syntax coloring, indentation, line numbers, and automatic creation of closing braces and quotes
- JavaScript support
- Script macros for common code shortcuts

This feature requires the Syntax editor plugin.
Script macro maintenance

Administrators can define new script macros or modify existing script macros.

Role required: admin

Script macros provide shortcuts for typing commonly used code. Several script macros are available by default. Administrators can define new or modify existing script macros.

1. Navigate to System Definition > Syntax Editor Macros.
2. Click New or select the macro to edit.
3. Define the macro details with the fields listed in the table below.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Macro keyword text users type to insert macro text.</td>
</tr>
<tr>
<td>Comments</td>
<td>Description of the macro. This text appears when the user types help.</td>
</tr>
<tr>
<td>Text</td>
<td>Full macro text that replaces the name in the editor.</td>
</tr>
</tbody>
</table>

Syntax editor plugin

Enable the syntax editor plugin to use the syntax editor.

The syntax editor enables the following features for all script fields:

- JavaScript syntax coloring, indentation, line numbers, and automatic creation of closing braces and quotes
- Code editing functions
- Code syntax checking
- Script macros for common code shortcuts
JavaScript syntax editor

The syntax editor can be disabled or enabled by modifying the `glide.ui.javascript_editor` property in the `sys_properties.list`. In addition, administrators can configure the syntax editor to show error and warning indicators next to a line of code that contains an error by modifying the `glide.ui.syntax_editor.show_warnings_errors` property. For information on the `sys_properties.list`, refer to Available system properties.

Note: Administrators can disable or enable the syntax editor for all users, regardless of user preference.

Searching for errors by line

To locate the exact position of the error in a large script, click the Go to line icon.

This feature is particularly useful when you are encountering a syntax error in a log file rather than in the ServiceNow record itself. In this case, you can navigate to the record and search for errors by line number. In the dialog box that appears, enter the line number of an error, and then click OK. Your view moves to the site of the error, and the cursor marks the correct line and column.

Note: For this feature to function, you must disable the Syntax Editor.
Navigate to a line number

When the syntax editor is disabled, users can navigate to a specific line in the code using the Go to line icon (computational result).

1. Click the Go to line icon
   ![Go to line icon](image)

   Note: This icon is not available when the editor is enabled.

2. Enter a number in the field and then press Enter.

Syntax editor JavaScript support

The syntax editor provides editing functions to support editing JavaScript scripts.

JavaScript editing functions

<table>
<thead>
<tr>
<th>Icon</th>
<th>Keyboard Shortcut</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>N/A</td>
<td>Toggle Syntax Editor</td>
<td>Disables the syntax editor. Click the button again to enable the syntax editor.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Access Key + R</td>
<td>Format Code</td>
<td>Applies the proper indentation to the script.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Access Key + C</td>
<td>Comment Selected Code</td>
<td>Comments out the selected code.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Access Key + U</td>
<td>Uncomment Selected Code</td>
<td>Removes comment codes from the selected code.</td>
</tr>
<tr>
<td>Icon</td>
<td>Keyboard Shortcut</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>🔄</td>
<td>N/A</td>
<td>Check Syntax</td>
<td>Checks the code for syntax errors. By default, the system automatically checks for syntax errors as you type in a script field. If an error or warning is found, the syntax editor displays a bullet beside the script line containing the error or warning. This check occurs on all script fields.</td>
</tr>
<tr>
<td>🔍</td>
<td>Access Key + \</td>
<td>Start Searching</td>
<td>Highlights all occurrences of a search term in the script field and locates the first occurrence. Click the icon, then enter the search term and press Enter. You can use regular expressions enclosed in slashes to define the search term. For example, the term /a(3)/ locates aaa.</td>
</tr>
<tr>
<td>🔄</td>
<td>Access Key + (</td>
<td>Find Next</td>
<td>Locates the next occurrence of the current search term in the script field. Use Start Searching to change the current search term.</td>
</tr>
<tr>
<td>🔍</td>
<td>Access Key + )</td>
<td>Find Previous</td>
<td>Locates the previous occurrence of the current search term in the script field. Use Start Searching to change the current search term.</td>
</tr>
<tr>
<td>Icon</td>
<td>Keyboard Shortcut</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><img src="110x723" alt="Image" /></td>
<td>Access Key + W</td>
<td>Replace</td>
<td>Replaces the next occurrence of a text string in the script field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. Click the icon, then enter the string to replace and press <strong>Enter</strong>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can use regular expressions enclosed in slashes to define the string</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to replace. For example, the term <code>/a{3}/</code> locates <code>aaa</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Enter the replacement string and press <strong>Enter</strong>.</td>
</tr>
<tr>
<td><img src="110x704" alt="Image" /></td>
<td>Access Key + ;</td>
<td>Replace All</td>
<td>Replaces all occurrences of a text string in the script field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. Click the icon, then enter the string to replace and press <strong>Enter</strong>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can use regular expressions enclosed in slashes to define the string</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to replace. For example, the term <code>/a{3}/</code> locates <code>aaa</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Enter the replacement string and press <strong>Enter</strong>.</td>
</tr>
<tr>
<td><img src="110x247" alt="Image" /></td>
<td>N/A</td>
<td>Save</td>
<td>Saves changes without leaving the current view. Use this button in full</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>screen mode to save without returning to standard form view.</td>
</tr>
<tr>
<td><img src="110x168" alt="Image" /></td>
<td>Access Key + L</td>
<td>Toggle Full Screen Mode</td>
<td>Expands the script field to use the full form view for easier editing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Click the button again to return to standard form view. This feature is not</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>available for Internet Explorer.</td>
</tr>
<tr>
<td>Icon</td>
<td>Keyboard Shortcut</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>![Help Icon]</td>
<td>Access Key + P</td>
<td>Help</td>
<td>Displays the keyboard shortcuts help screen.</td>
</tr>
</tbody>
</table>

**JavaScript editing tips**

- To fold a code block, click the minus sign beside the first line of the block. The minus sign only appears beside blocks that can be folded. To unfold the code block, click the plus sign.
- To insert a fixed space anywhere in your code, press Tab.
- To indent a single line of code, click in the leading white space of the line and then press Tab.
- To indent one or more lines of code, select the code and then press Tab. To decrease the indentation, press Shift + Tab.
- To remove one tab from the start of a line of code, click in the line and press Shift + Tab.

**JavaScript resources**

Scripts use ECMA 262 standard JavaScript. Helpful resources include:

- Mozilla: [http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference)
- History and overview: [http://javascript.crockford.com/survey.html](http://javascript.crockford.com/survey.html)

**Syntax editor macros**

Script macros provide shortcuts for typing commonly used code. To insert macro text into a script field, enter the macro keyword followed by the Tab.

**vargr**

- **Description**: Inserts a standard GlideRecord query for a single value.
- **Output**:

```javascript
var gr = new GlideRecord('');
gr.addQuery('name', 'value');
gr.query();
if (gr.next()) {
}
```

**vargror**

- **Description**: Inserts a GlideRecord query for two values with an OR condition.
- **Output**:

```javascript
var gr = new GlideRecord('');
var qc = gr.addQuery('field', 'value1');
```
for

- **Description**: Inserts a standard recursive loop with an array.
- **Output**:

```javascript
for (var i=0; i< myArray.length; i++) {
    //myArray[i];
}
```

info

- **Description**: Inserts a GlideSystem information message.
- **Output**:

```javascript
gs.addInfoMessage('"");
```

method

- **Description**: Inserts a blank JavaScript function template.
- **Output**:

```javascript
/*_________________________________________________________________
* Description:
* Parameters:
* Returns:
*________________________________________________________________*/
: function() {
    },
```

doc

- **Description**: Inserts a comment block for describing a function or parameters.
- **Output**:

```javascript
/**
 * Description:
 * Parameters:
 * Returns:
 */
```
Script syntax error checking

All script fields provide controls for checking the syntax for errors and for locating the error easily when one occurs. The script editor places the cursor at the site of a syntax error and lets you search for errors in scripts by line number.

Script syntax check

The script editor notifies you of syntax errors in your scripts in the following situations.

- Save a new record or update an existing record. A banner appears at the bottom of the editor showing the location of the first error (line number and column number), and the cursor appears at the site of the error. Warnings presented at Save or Update show only one error at a time.

- Click the syntax checking icon before saving or updating a record. A banner appears at the bottom of the editor showing the location of all errors in the script, and the cursor appears at the site of the first error.
Script syntax error

Syntax editor keyboard shortcuts and actions

The syntax editor offers keyboard shortcuts and actions to assist in writing code.

<table>
<thead>
<tr>
<th>Keyboard shortcut or action</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyboard shortcut or action</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| Scripting assistance | Displays a list of valid elements at the insertion point such as:  
- Class names  
- Function names  
- Object names  
- Variable names  
Double-click an entry to add it to the script. | ![Script Example](image) |
<table>
<thead>
<tr>
<th>Keyboard shortcut or action</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter a period character after a valid class name.</td>
<td>Displays a list methods for the class. Double-click an entry to add it to the script.</td>
<td></td>
</tr>
<tr>
<td>Enter an open parenthesis character after a valid class, function, or method name.</td>
<td>Displays the expected parameters for the class or method. Enter the expected parameters as needed.</td>
<td></td>
</tr>
<tr>
<td>Toggle full screen mode</td>
<td>Switches between displaying the form with the full screen and displaying it normally.</td>
<td></td>
</tr>
<tr>
<td>Keyboard shortcut or action</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Format code</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• <strong>Windows</strong>: Control+Shift+B</td>
<td>Formats the selected lines to improve readability.</td>
<td><img src="image1" alt="Example" /></td>
</tr>
<tr>
<td>• <strong>Mac</strong>: Command+Shift+B</td>
<td></td>
<td><img src="image2" alt="Example" /></td>
</tr>
<tr>
<td>Keyboard shortcut or action</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>Toggle comment</td>
<td>Add or remove the comment characters <code>//</code> from the selected lines.</td>
<td><img src="image" alt="Example" /></td>
</tr>
<tr>
<td><strong>Windows</strong>: Control+/</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mac</strong>: Command+/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyboard shortcut or action</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>Insert macro text</td>
<td>Inserts macro text at the current position.</td>
<td>![Example Image]</td>
</tr>
</tbody>
</table>

1. In the **Script** field, type the macro keyword text. For example `help`.
2. Press Tab.

Search
<table>
<thead>
<tr>
<th>Keyboard shortcut or action</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start search</strong>&lt;br&gt;• <strong>Windows:</strong> Control+F&lt;br&gt;• <strong>Mac:</strong> Command+F</td>
<td>Highlights all occurrences of a search term in the script field and locates the first occurrence. You can create <em>regular expressions</em> by enclosing the search terms between slash characters. For example, the search term <code>/a\(3\)/</code> locates the string <code>aaa</code>.</td>
<td>![Example Image]</td>
</tr>
<tr>
<td>Keyboard shortcut or action</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Find next</td>
<td>Locates the next occurrence of the current search term in the script field. Use <strong>Start Searching</strong> to change the current search term.</td>
<td><img src="example.png" alt="Example" /></td>
</tr>
<tr>
<td><strong>Windows:</strong> Control+G</td>
<td></td>
<td><img src="example.png" alt="Example" /></td>
</tr>
<tr>
<td><strong>Mac:</strong> Command+G</td>
<td></td>
<td><img src="example.png" alt="Example" /></td>
</tr>
</tbody>
</table>

<p>| Find previous              | Locates the previous occurrence of the current search term in the script field. Use <strong>Start Searching</strong> to change the current search term. | <img src="example.png" alt="Example" /> |
| <strong>Windows:</strong> Control+Shift+G |                                                                              | <img src="example.png" alt="Example" /> |
| <strong>Mac:</strong> Command+Shift+G   |                                                                              | <img src="example.png" alt="Example" /> |</p>
<table>
<thead>
<tr>
<th>Keyboard shortcut or action</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace</td>
<td>Replaces the next occurrence of a text string in the script field.</td>
<td><img src="image" alt="Example" /></td>
</tr>
<tr>
<td>Windows: Control+E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mac: Command+E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>Keyboard shortcut or action</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace all</td>
<td>Replaces all occurrences of a text string in the script field.</td>
<td>![Example Image]</td>
</tr>
<tr>
<td><strong>Windows:</strong> Control+;</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mac:</strong> Command+;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Help</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyboard shortcut or action</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>Help</td>
<td>Displays the list of syntax editor keyboard shortcuts.</td>
<td><img src="image" alt="Editor Key Map" /></td>
</tr>
<tr>
<td>· <strong>Windows</strong>: Control+H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>· <strong>Mac</strong>: Command+H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show description</td>
<td>Displays API documentation for the scripting element at the cursor's current location.</td>
<td><img src="image" alt="Show Description" /></td>
</tr>
<tr>
<td>· <strong>Windows</strong>: Control+J</td>
<td></td>
<td></td>
</tr>
<tr>
<td>· <strong>Mac</strong>: Command+J</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyboard shortcut or action</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>Show macros</td>
<td>Displays the list of available syntax editor macros as text within the script field.</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

1. In the **Script** field, type `help`.
2. Press Tab.

### Business rules

A business rule is a server-side script that runs when a record is displayed, inserted, updated, or deleted, or when a table is queried.

Use business rules to accomplish tasks like automatically changing values in form fields when certain conditions are met, or to create events for email notifications and script actions.

**Note:** Business rules can make use of scripts to take actions on records in the database. However, there are several other scripting options available on the platform, such as client scripts and UI actions.

### How business rules work

To configure business rules, you first need to determine when the business rule should run and what action it should take.

### When business rules run

Business rules run based on two sets of criteria:
- The time that the business rule is configured to run relative to a record being modified or accessed.
- The database operation that the system takes on the record.

The following options are provided to determine the time the business rule should run:

<table>
<thead>
<tr>
<th>Option</th>
<th>When the rule runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before</td>
<td>After the user submits the form but before any action is taken on the record in the database.</td>
</tr>
<tr>
<td>After</td>
<td>After the user submits the form and after any action is taken on the record in the database.</td>
</tr>
<tr>
<td>Async</td>
<td>When the scheduler runs the scheduled job created from the business rule. The system creates a scheduled job from the business rule after the user submits the form and after any action is taken on the record in the database.</td>
</tr>
<tr>
<td>Display</td>
<td>Before the form is presented to the user, just after the data is read from the database.</td>
</tr>
</tbody>
</table>

**Note:** Asynchronous business rules do not have access to the previous version of a record. Therefore, the `changes()`, `changesTo()`, and `changesFrom()` `GlideElement` methods do not work with async rule script. However, the condition builder and condition field (advanced view) both support the `changes()`, `changesTo()`, and `changesFrom()` methods.

The following options are provided to determine the database operation that the system takes on the record:

<table>
<thead>
<tr>
<th>Option</th>
<th>When the rule runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert</td>
<td>When the user creates a new record and the system inserts it into the database.</td>
</tr>
<tr>
<td>Update</td>
<td>When the user modifies an existing record.</td>
</tr>
<tr>
<td>Query</td>
<td>Before a query for a record or list of records is sent to the database. Typically you should use query for before business rules. See Before-Query example.</td>
</tr>
<tr>
<td>Delete</td>
<td>When the user deletes a record.</td>
</tr>
</tbody>
</table>

This image shows when different types of business rules run:
Note: Business rules apply consistently to records regardless of whether they are accessed through forms, lists, or web services. This is one major difference between business rules and client scripts, which apply only when the form is edited.

Business rule actions

Business rules can perform a variety of actions. Common types of actions are:

- Changing field values on a form that the user is updating. Field values can be set to specific values available for that field, values copied from other fields, and relative values determined by the user’s role.
- Displaying information messages to the user.
- Changing values of child tasks based on changes to parent tasks.
- Preventing users from accessing or modifying certain fields on a form.
- Aborting the current database transaction. For example, if certain conditions are met, prevent the user from saving the record in the database.

Administrators can set field values, create information messages, and abort transactions without writing a script.
Prevent recursive business rules

Avoid using `current.update()` in a business rule script. The `update()` method triggers business rules to run on the same table for insert and update operations, leading to a business rule calling itself over and over. Changes made in before business rules are automatically saved when all before business rules are complete, and after business rules are best used for updating related, not current, objects. When a recursive business rule is detected, the system stops it and logs the error in the system log. However, `current.update()` causes system performance issues and is never necessary.

You can prevent recursive business rules by using the `setWorkflow()` method with the `false` parameter. The combination of the `update()` and `setWorkflow()` methods is only recommended in special circumstances where the normal before and after guidelines mentioned above do not meet your requirements.

Business rules in scoped applications

Every business rule is assigned to either a private application scope or to the global scope.

The types of business rules you can create and how you can access those rules varies depending on the scope of the business rule and the scope of the table it runs on.

Note: The term `global` can refer to two different aspects of a business rule: the table it runs on and the scope it runs in. Business rules can either run on specific tables or be global. In addition, they can be in the global scope or in a private application scope.

Business rules on specific tables

Most business rules run on a specific table, which is defined in the `Table` field. You can create business rules on tables in the same scope and on tables that allow configuration records from another application scope.

For tables that are in a different scope than the business rule record, the types of rules are limited.

- You can create a rule where **When is async** with any of the following options:
  - *Insert*, *Update*, and *Delete* database operations. You cannot select *Query*.
  - *Set field values* actions and scripts (the `Script` field).

- You can create a rule where **When is before** with any of the following options:
  - *Insert*, *Update*, and *Delete* database operations. You cannot select *Query*.
  - *Set field values* actions only. You cannot write scripts and you cannot abort the database transaction.

- You cannot create any other types of business rules on tables in a different scope.

Business rules on specific tables cannot be accessed by other business rules or scripts.

Global business rules

Warning: Consider using script includes instead of global business rules. Script includes load only on request while global business rules load on every page in the system.
Global business rules are business rules where the Table field is set to Global. Global business rules may be accessible on multiple tables and from other scripts, depending on their scope protection. For a global business rule, define the scope protection by setting the Accessible from field:

- **This application scope only**: prevents applications in a different scope than the business rule from calling this business rule.
- **All application scopes**: allows any application to call this business rule.

**Note:** Global business rules do not support domain separation.

**Scripts in scoped business rules**

When you write a script in a business rule, you can access:

- Any script includes and global business rules in the same scope as the business rule.
- Script includes and global business rules that allow applications in a different scope to call them. To call functions from another scope, you must specify the scope of the function.
- For business rules in a unique scope, you can access the scoped system APIs only.

For more information on scoped applications, see scoped application scripting on the developer portal.

**Create a business rule**

You can create any type of business rule to run when a record is displayed, inserted, updated, or deleted, or when a table is queried.

1. Navigate to System Definition > Business Rules.
2. Click New.
3. Fill in the fields, as appropriate.

**Note:** You might need to configure the form to see all fields.

**Business Rule fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a name for the business rule.</td>
</tr>
<tr>
<td>Table</td>
<td>Select the table that the business rule runs on.</td>
</tr>
<tr>
<td>Application</td>
<td>Application that contains this business rule.</td>
</tr>
</tbody>
</table>

**Note:** The list shows only tables and database views that meet the scope protections for business rules. Business rules defined for a database view can run only on Query. A business rule for a database view cannot run on insert, update, or delete.
<table>
<thead>
<tr>
<th><strong>Field</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessible from</td>
<td>Scope protection for a global business rule.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This field is visible only when the <strong>Table</strong> field is set to <strong>Global</strong>. It does not apply to rules that run on specific tables.</td>
</tr>
<tr>
<td>Active</td>
<td>Select this check box to enable the business rule.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Select this check box to see the advanced version of the form.</td>
</tr>
<tr>
<td>When</td>
<td><strong>(Advanced)</strong> Select when this business rule should execute: <strong>display</strong>, <strong>before</strong>, <strong>async</strong>, or <strong>after</strong> the database operation is complete.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Consider setting the <strong>Priority</strong> for <strong>async</strong> business rules as the system uses this value when creating the associated scheduled job.</td>
</tr>
<tr>
<td>Order</td>
<td><strong>(Advanced)</strong> Enter a number indicating the sequence in which this business rule should run. If there are multiple rules on a particular activity, the rules run in the order specified here, from lowest to highest.</td>
</tr>
<tr>
<td>Insert</td>
<td>Select this check box to execute the business rule when a record is inserted into the database.</td>
</tr>
<tr>
<td>Update</td>
<td>Select this check box to execute the business rule when a record is update.</td>
</tr>
<tr>
<td>Delete</td>
<td><strong>(Advanced)</strong> Select this check box to execute the business rule when a record is deleted from the database.</td>
</tr>
<tr>
<td>Query</td>
<td><strong>(Advanced)</strong> Select this check box to execute the business rule when a table is queried.</td>
</tr>
<tr>
<td>Filter Conditions</td>
<td>Use the condition builder to determine when the business rule should run based on the field values in the selected Table. You can also use the <strong>Condition</strong> field to build a condition with a script.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Filters based on string compares are case-sensitive.</td>
</tr>
<tr>
<td>Role Conditions</td>
<td>Select the roles that users who are modifying records in the table must have for this business rule to run.</td>
</tr>
<tr>
<td>Actions</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Set field values   | Set values for fields in the selected Table using the choice lists:  
- The field  
- The assignment operator:  
  - **To:** An exact value  
  - **Same as:** The value of another field  
  - **To (dynamic):** A value relative to the user configuring the business rule or a user with a specific role  
- The value                                                                                                                                                                                                                     |
| Add message        | Select this check box and enter a message that appears when this business rule is run                                                                                                                                                                                                                                                                                                                                                      |
| Abort action       | Select this check box to abort the current database transaction. For example, on a before insert business rule, if the conditions are met, do not insert the record into the database.  
If you select this option, you cannot perform additional actions on the record, such as setting field values and running scripts. You can still display a message to users by selecting the **Add message** check box and composing the message. |
| Advanced           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Condition          | Create a JavaScript conditional statement to specify when the business rule should run. By adding the condition statement to this field, you tell the system to evaluate the condition separately and run the business rule only if the condition is true. If you decide to include the condition statement in the **Script** field or if you use the condition builder, leave this field blank. To have the instance reevaluate the condition statement a second time before running an async business rule, add the system property **glide.businessrule.async_condition_check** and set the value to true. |
### Field Description

**Script**

(Advanced) Create a script that runs when the defined condition is true. The system automatically populates this field with a function name that matches the *When* value.

- `onAfter`
- `onAsync`
- `onBefore`
- `onDisplay`

**Note:** The function name must match the *When* value.

For more information and examples, see Scripting in Business Rules.

**Related list: Versions**

**Versions**

Shows all versions of the business rule. Use this list to compare versions or to revert to a previous version.

### Global variables in business rules

Predefined global variables are available for use in business rules.

Use the following predefined global variables to reference the system in a business rule script.

<table>
<thead>
<tr>
<th>Global variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>current</code></td>
<td>The current state of the record being referenced. <em>Check for null</em> before using this variable.</td>
</tr>
<tr>
<td><code>previous</code></td>
<td>The state of the referenced record prior to any updates made during the execution context, where the execution context begins with the first update or delete operation and ends after the script and any referenced business rules are executed. If multiple updates are made to the record within one execution context, <code>previous</code> will continue to hold the state of the record before the first update or delete operation. Available on update and delete operations only. Not available on asynch operations. <em>Check for null</em> before using this variable.</td>
</tr>
<tr>
<td><code>g_scratchpad</code></td>
<td>The scratchpad object is available on display rules, and is used to pass information to the client to be accessed from client scripts.</td>
</tr>
<tr>
<td><code>gs</code></td>
<td>References to functions.</td>
</tr>
</tbody>
</table>

The variables `current`, `previous`, and `g_scratchpad` are global across all business rules that run for a transaction.

---

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Prevent null pointer exceptions

In some cases, there may not be a current or previous state for the record when a business rule runs, which means that the variables will be null. To check for null before using a variable, add the following code to your business rule:

```java
if (current == null) // to prevent null pointer exceptions.
    return;
```

Define variables

User-defined variables are globally scoped by default. If a new variable is declared in an order 100 business rule, the business rule that runs next at order 200 also has access to the variable. This may introduce unexpected behavior.

To prevent such unexpected behavior, always wrap your code in a function. This protects your variables from conflicting with system variables or global variables in other business rules that are not wrapped in a function. Additionally, variables such as current must be available when a function is invoked in order to be used.

The following script is vulnerable to conflicts with other code. If the variable gr is used in other rules, the value of the variable may unexpectedly change.

```java
var gr = new GlideRecord('incident');
gr.query();
while(gr.next()) {
    //do something
}
```

When this script is wrapped in a function, the variable is available only within the function and does not conflict with other functions using a variable named gr.

```java
myFunction();
function myFunction() {
    var gr = new GlideRecord('incident');
gr.query();
while(gr.next()) {
    //do something
}
}
```

Display business-rules

Display rules are processed when a user requests a record form.

The data is read from the database, display rules are executed, and the form is presented to the user. The current object is available and represents the record retrieved from the database. Any field changes are temporary since they are not yet submitted to the database. To the client, the form values appear to be the values from the database; there is no indication that the values were modified from a display rule. This is a similar concept to calculated fields.

The primary objective of display rules is to use a shared scratchpad object, g_scratchpad, which is also sent to the client as part of the form. This can be useful when you need to build client scripts that require server data that is not typically part of the record being displayed. In most cases, this would require a client script making a call back to the server. If the data can be determined prior to the form being displayed, it is more efficient to provide the data to the client on the initial load.
The form scratchpad object is an empty object by default, and used only to store name:value pairs of data.

To populate the form scratchpad with data from a display rule:

```javascript
// From display business rule
g_scratchpad.someName = "someValue";
g_scratchpad.anotherName = "anotherValue";

// If you want the client to have access to record fields not being displayed on the form
g_scratchpad.created_by = current.sys_created_by;

// These are simple examples, in most cases you'll probably perform some other queries to test or get data
```

To access the form scratchpad data from a client script:

```javascript
// From client script
if(g_scratchpad.someName == "someValue") {
    // do something special
}
```

**Task Active State Management business rule**

This business rule determines whether the active field value needs to change based on changes to the **State** field.

The Task Active State Management business rule is executed when the **State** is changed for a task record. Its execution order is 50, and it runs before most other task business rules.

If the current task table has the **close_states** attribute defined on its table, or if it is inherited from a higher-level table, then the rule determines whether the active field needs to change. This is done by comparing the previous and current state values.

- If the state changes from an active state to an inactive state, the **Active** field is set to false.
- If the state changes from an inactive state to an active state, the **Active** field is set to true, effectively re-activating or re-opening the task.

It is recommended that you leverage the `(current.active.changesTo(true/false))` action in your business rule, as opposed to creating rules on each task table that mark tasks as inactive or active.

**Example business rule scripts**

Find an example business rule script that helps you with a requirement of your organization.

**Compare date fields in a business rule**

It is possible to compare two date fields or two date and time fields in a business rule, and abort a record insert or update if they are not correct.

For example, you may want a start date to be before an end date. The following is an example script:

```javascript
if (!current.u_date1.nil() && !current.u_date2.nil()) {
    var start = current.u_date1.getGlideObject().getNumericValue();
    var end = current.u_date2.getGlideObject().getNumericValue();
    if (start > end) {
        gs.addInfoMessage('start must be before end');
        current.u_date1.setError('start must be before end') ;
        current.setAbortAction(true);
```
This example has been tested in global scripts, and may need changes to work in scoped scripts. In addition to possibly needing API changes, security is more strict in scoped scripts.

As a good practice, make the business rule a before rule for insert and update actions. In the example script:

- `u_date1` and `u_date2` are the names of the two date fields. Replace these names with your own field names.
- The first line checks that both fields actually have a value.
- The next two lines create variables that have the dates' numerical values.
- The next two lines create different alert messages for the end user: one at the top of the form and one by the `u_date1` field in the form.
- The last line aborts the insert or update if the date fields are not correct.

Here is a more complex example of the above comparison. If you have more than one pair of start and end dates, you can use arrays as shown. Additionally, this script requires the input dates to be within a certain range, in this case, no fewer than 30 days in the past and no more than 365 days in the future.

```javascript
// Enter all start and end date fields you wish to check, as well as the previous values
// Make sure that you keep the placement in the sequence the same for all pairs
var startDate = new Array(current.start_date,current.work_start);
var prevStartDate = new Array(previous.start_date,previous.work_start);
var endDate = new Array(current.end_date,current.work_end);
var prevEndDate = new Array(previous.end_date,previous.work_end);

// The text string below is added to the front of ' start must be before end'
var userAlert = new Array('Planned','Work');

// Set the number of Previous Days you want to check
var pd = 30;
// Set the number of Future Days you want to check
var fd = 365;

// You shouldn't have to modify anything below this line
var nowdt = new GlideDateTime();
nowdt.setDisplayValue(gs.nowDateTime());
var nowMs = nowdt.getNumericValue();
var pdms = nowMs;
// Subtract the product of previous days to get value in milliseconds
pdms -= pd * 24 * 60 * 60 * 1000;
var fdms = nowMs;
// Add the product of future days to get value in miliseconds
fdms += fd * 24 * 60 * 60 * 1000;
var badDate = false;

// Iterate through all start and end date / time fields
for (x = 0; x < startDate.length; x ++) {
  if (!(startDate[x].nil()) && (!endDate[x].nil())) {
    var start = startDate[x].getGlideObject().getNumericValue();
    var end = endDate[x].getGlideObject().getNumericValue();
    if (start > end) {
      gs.addInfoMessage(userAlert[x] + ' start must be before end');
    }
  }
}
```
```javascript
startDate[x].setError(userAlert[x] + ' start must be before end');
badDate = true;
} else if ((prevStartDate[x]) != (startDate[x])) {
    if (start < pdms) {
        gs.addInfoMessage(userAlert[x] + ' start must be fewer than ' + pd + ' days ago');
        startDate[x].setError(userAlert[x] + ' start must be fewer than ' + pd + ' days ago');
        badDate = true;
    } } else if ((prevEndDate[x]) != (endDate[x])) {
    if (end > fdms) {
        gs.addInfoMessage(userAlert[x] + ' end must be fewer than ' + fd + ' days ahead');
        endDate[x].setError(userAlert[x] + ' end must be fewer than ' + fd + ' days ahead');
        badDate = true;
    } } } } 
if (badDate == true) {
    current.setAbortAction(true);
}
```

**Parse XML payloads**

Fields in XML format can be parsed with the system's `getXMLText` function.

Fields that get inserted into the database in XML format, such as the payload of an `ecc_event` row, can be parsed with the system's `getXMLText` function. The `getXMLText` function takes a string and an XPATH expression. For example:

```javascript
var name = gs.getXMLText("<name>joe</name>", "/name");
```

returns the string 'joe'.

Assuming that the field "payload" contains XML, the function call might look like:

```javascript
var name = gs.getXMLText(current.payload, "/name");
```

For information on XPATH, visit [w3schools](https://www.w3schools.com).

**Abort a database action in a before business-rule**

In a before business rule script, you can cancel or abort the current database action using the `setAbortAction()` method.

For example, if the before business rule is executed during an insert action, and you have a condition in the script that calls `current.setAbortAction(true)`, the new record stored in `current` is not created in the database. The business rule continues to run after calling `setAbortAction()` and all subsequent business rules will execute normally. Calling this method only prevents the database action from occurring.

You can use the `isActionAborted()` method to determine if the current database action (insert, update, delete) is going to be aborted. `isActionAborted()` is initialized for new threads and the `next()` method explicitly sets its value to false.

**Note:** `setAbortAction()` can only be executed from the same scope as the record whose action is being aborted. `current.setAbortAction()` is not honored if executed in a business rule that is defined in a different scope.

**Determine the operation that triggered the business rule**

You can write a script for a business rule that is triggered on more than one database action.
If you want the business rule script to dynamically branch depending on the action that triggered the event, you can use the `operation()` function. For example:

```plaintext
if(current.operation() == "update") {
    current.updates ++;
} else if(current.operation() == "insert") {
    current.updates = 0;
}
```

### Use an OR condition in a business rule

An **OR** condition can be added to any query part within a business rule.

An **OR** condition can be added to any query part within a business rule with the `addOrCondition()` method. The example below shows a query for finding all the incidents that have either a 1 or a 2 priority. The first `addQuery()` condition is defined as a variable and is used in the **OR** condition.

```plaintext
var inc = new GlideRecord('incident');
var qc = inc.addQuery('priority','1');
qc.addOrCondition('priority','2');
inc.query();
while(inc.next()) {
    // processing for the incident goes here
}
```

The following script is a more complex example, using two query condition variables doing the equivalent of `priority = 1 OR priority = 2` **AND** `impact = 2 OR impact = 3`. The results of the **OR** condition are run with two variables, `qc1` and `qc2`. This allows you to manipulate the query condition object later in the script, such as inside an **IF** condition or **WHILE** loop.

```plaintext
var inc = new GlideRecord('incident');
var qc1 = inc.addQuery('priority','1');
qc1.addOrCondition('priority','2');
var qc2 = inc.addQuery('impact','2');
qc2.addOrCondition('impact','3');
inc.query();
while(inc.next()) {
    // processing for the incident goes here
}
```

### Reference a Glide list from a business rule

A field defined as a glide list is an array of values stored in a single field.

Here are some examples of how to process a glide_list field when writing business rules. Generally a glide_list field contains a list of reference values to other tables.

### Examples

For example, the **Watch list** field within tasks is a glide_list containing references to user records. The code below shows how to reference the field.

```plaintext
// list will contain a series of reference (sys_id) values separated by a comma
// array will be a javascript array of reference values
var list = current.watch_list.toString();
var array = list.split(",");
for (var i=0; i < array.length; i++) {
    gs.print("Reference value is: " + array[i]);
}
```
You can also get the display values associated with the reference values by using the `getDisplayValue()` method as shown below.

```javascript
// list will contain a series of display values separated by a comma
// array will be a javascript array of display values
var list = current.watch_list.getDisplayValue();
var array = list.split(",");
for (var i=0; i < array.length; i++) {
  gs.print("Display value is: " + array[i]);
}
```

**Use `indexOf("searchString")` to find a string in a Glide list**

Use `indexOf("searchString")` to return the location of the string passed into the method if the glide list field, such as a Watch list, has at least one value in it.

If the field is empty, it returns `undefined`. To avoid returning an undefined value, do any of the following:

- Force the field to a string, such as: `watch_list.toString().indexOf("searchString")`
- Check for an empty Glide list field with a condition before using `indexOf()`, such as: if `(watch_list.nil() || watch_list.indexOf("searchString") == -1)`

**Lock user accounts**

You can lock user accounts if the user is not active.

The following business rule script locks user accounts if the user is not active in the LDAP directory or the user does not have self-service, itil, or admin access to the instance.

```javascript
// Lock accounts if bcNetIDStatus != active in LDAP and user does not
// have self-service, itil or admin role
var rls = current.accumulated_roles.toString();
if(current.u_bcnetidstatus == 'active' && (rls.indexOf(',itil,') > 0 ||
  rls.indexOf(',admin,') > 0 ||
  rls.indexOf(',ess,') > 0 )) {
  current.locked_out = false;
}
else {
  current.locked_out = true;
}
var gr = new GlideRecord("sys_user");
gr.query();
while(gr.next()) {
  gr.update();
  gs.print("updating " + gr.getDisplayValue());
}
```

**Default before-query business rule**

You can use a query business rule that executes before a database query is made.

Use this query business rule to prevent users from accessing certain records. Consider the following example from a default business rule that limits access to incident records.

- **Name**: incident query
- **Table**: Incident
- **When**: before, query
- **Script**:

  ```javascript
  if(!gs.hasRole("itil") && gs.isInteractive()) {
    var u = gs.getUserID();
    var qc =
      current.addQuery("caller_id",u).addOrCondition("opened_by",u).addOrCondition("watch_list"...}
  ```
gs.print("query restricted to user: " + u); }

This example prevents users from accessing incident records unless they have the itil role, or are listed in the Caller orOpened by field. So, for example, when self-service users open a list of incidents, they can only see the incidents they submitted.

Note: You can also use access controls to restrict the records that users can see.

Server scripts

Server scripts run on the server or database. They can change the appearance or behavior of ServiceNow or run as business rules when records and tables are accessed or modified.

Server-side Glide APIs (Application Programming Interfaces) provide classes and methods that you can use in scripts to perform server-side tasks.

Immediately invoked function expressions

An immediately invoked function expression (IIFE) is both declared and invoked within the same script field.

The system uses immediately invoked function expressions when a script runs in a single context, such as in a transform map script. Functions that run from multiple contexts use Script includes instead.

By enclosing a script in an immediately invoked function expression, you can:

- Ensure that the script does not impact other areas of the product, such as by overwriting global variables.
- Pass useful variables or objects as parameters.
- Identify function names in stack traces.
- Eliminate having to make separate function calls.

An immediately invoked function expression follows this format:

```
(function functionName(parameter){
   //The script you want to run
 })('value'); //Note the parenthesis indicating this function should run.
```

You can declare functions within the immediately invoked function expression. These inner functions are accessible only from within the immediately invoked function expression.

```
(function functionName(parameter){
   function helperFunction(parameter){ //return some value
      var value = helperFunction(parameter); //Valid function call.
      //perform any other script actions
   })('value');
   var value2 = helperFunction(parameter); //Invalid. This function is not accessible from outside the self-executing function.
```
Script includes

Script includes are used to store JavaScript that runs on the server.

Create script includes to store JavaScript functions and classes for use by server scripts. Each script include defines either an object class or a function.

Consider using script includes instead of global business rules because script includes are only loaded on request.

Script include form

Script includes have a name, description and script. They also specify whether they are active or not, and whether they can be called from a client script.

To access script includes, navigate to System Definitions > Script Includes.

Script include form

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the script include. If you are defining a class, this must match the name of the class, prototype, and type. If you are using a classless (on-demand) script include, the name must match the function name.</td>
</tr>
<tr>
<td>Client callable</td>
<td>Makes the script include available to client scripts, list/report filters, reference qualifiers, or if specified as part of the URL.</td>
</tr>
<tr>
<td>Application</td>
<td>The application where this script include resides.</td>
</tr>
<tr>
<td>Accessible from</td>
<td>Sets which applications can access this script include:</td>
</tr>
<tr>
<td></td>
<td><strong>All application scopes</strong></td>
</tr>
<tr>
<td></td>
<td>Can be accessed from any application scope.</td>
</tr>
<tr>
<td></td>
<td><strong>This application scope only</strong></td>
</tr>
<tr>
<td></td>
<td>Can be accessed only from the current application scope.</td>
</tr>
<tr>
<td>Active</td>
<td>Enables the script include when selected. Uncheck the active field to disable the script include.</td>
</tr>
<tr>
<td>Description</td>
<td>Provides descriptive content regarding the script include.</td>
</tr>
<tr>
<td>Script</td>
<td>Defines the server side script to run when called from other scripts. The script must define a single JavaScript class or a global function. The class or function name must match the Name field.</td>
</tr>
<tr>
<td>Package</td>
<td>The package that contains this script include.</td>
</tr>
<tr>
<td>Created by</td>
<td>The user who created this script include.</td>
</tr>
<tr>
<td>Updated by</td>
<td>The user who most recently updated this script include.</td>
</tr>
</tbody>
</table>
### Field: Protection policy

**Description:**

- **None**: Allows anyone to read and edit this downloaded or installed script include.
- **Read-only**: Allows anyone to read values from this downloaded or installed script include. No one can change script values on the instance on which they download or install the script include.
- **Protected**: Provides intellectual property protection for application developers. Customers who download the script include cannot see the contents of the script field. The script is encrypted in memory to prevent unauthorized users from seeing it in plain text.

### Related lists on the form view:

- **Versions**: Shows all versions of the script include. Use this list to compare versions or to revert to a previous version. See [Versions](#).

---

**Use script includes**

Script includes are found under System Definition or System UI. You can call existing script includes from a script.

To create an entirely new script include, you can follow the format of any of the existing script includes. In the example, the name of your Script Include is 'NewInclude' and there is a single function called 'myFunction'. It is important that the name of the script include match the name of the class, prototype, and type. When you create a new script include and give it a name, the system provides you a code snippet with the class and prototype set up properly.

```javascript
var NewInclude = Class.create();
NewInclude.prototype = {
  initialize: function() {},
  myFunction: function() { //Put function code here,
    type: 'NewInclude';
  }
};
```

You could then use the 'myFunction' line like this:

```javascript
var foo = new NewInclude();
foo.myFunction();
```

**Note:** Try not to modify a ServiceNow supplied script include. If you want a script include that does something similar to an existing one, copy it and make changes to the copy or consider extending the object. This is a common practice when using GlideAjax.
Privacy settings
The privacy setting for a client-callable script-include can be public or private. Most client-callable script-includes are marked private by default.

The private privacy-setting means that guests who access public pages cannot access the client-callable script-include. A private script cannot be executed by a non-logged-in user.

A public privacy-setting means that the client script can be executed by non-logged-in users that create an appropriate HTTP request. This can create a security problem if the client script provides confidential information.

The following script includes remain public by default because public pages need to access them:

- GlideSystemAjax
- SysMessageAjax
- KnowledgeMessagingAjax
- KnowledgeAjax
- PasswordResetAjax

Change privacy on all client-callable script includes
How to change the privacy on all client-callable script includes.

To provide further control over all client-callable script includes, administrators can add the property glide.script.ccsi.ispublic. This property changes the visibility of client-callable script includes by making them all public or private. Configure the property as follows:

<table>
<thead>
<tr>
<th>Title</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>glide.script.ccsi.ispublic</td>
</tr>
<tr>
<td>Type</td>
<td>true</td>
</tr>
<tr>
<td>Value</td>
<td>false</td>
</tr>
</tbody>
</table>
Client-callable script includes public property

**Change privacy on a single client callable script include**
Change the privacy setting for a single client-callable script include by adding the `isPublic()` function.

The `isPublic()` setting takes precedence over the `glide.script.ccsi.ispublic` property. For example, if the property is set to false making all client-callable script-includes private, and a script sets `isPublic()` to true, the script is public.

To change the privacy for a single client-callable script include, add the following method to the script include:

```javascript
isPublic:function(){return[true/false];},
```

An example to make the client script `NewInclude` private.

```javascript
var NewInclude =Class.create();
NewInclude.prototype={
    initialize:function(){},
```

```javascript
```
**UI pages**

UI pages can be used to create and display forms, dialogs, lists and other UI components. Use UI pages as widgets on dashboards. To find the UI pages, navigate to **System UI > UI Pages**. UI Pages use HTML, Jelly, and AngularJS. To learn more about creating an AngularJS UI page, see the *Building Apps with AngularJS* training.
ServiceNow    Kingston    Now Platform Custom Business Applications

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
The UI page form provides the following fields:

### UI page

<table>
<thead>
<tr>
<th>Field</th>
<th>Input Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name used to invoke the page via a URL (must not contain spaces).</td>
</tr>
<tr>
<td>HTML</td>
<td>Main component of the page, and it defines what will be rendered when the page is shown. It can contain either static X-HTML or dynamically generated content defined as Jelly, and it can call script includes and UI Macros.</td>
</tr>
<tr>
<td>Client Script</td>
<td>Client-side JavaScript that runs in the browser (e.g., functions called by buttons, etc.). It is intended to handle any client-side processing needed, for example setting focus to a field, or other interactive DHTML features after a page is loaded. Ultimately, a UI page's Client Scripts are deployed to the browser within a <code>&lt;script/&gt;</code> tag, so it could be defined within the page's HTML field to achieve the same effect. Using the Client Script field instead to define these scripts makes things much more tidy and readable though, and it keeps the Jelly and HTML from becoming unmanageable.</td>
</tr>
<tr>
<td>Processing Script</td>
<td>Script that runs on the server when the page is submitted. This is useful if your page has a form (defined with the <code>&lt;g:ui_form/&gt;</code> or <code>&lt;g:form/&gt;</code> tags).</td>
</tr>
</tbody>
</table>

Related lists on the form view:

<table>
<thead>
<tr>
<th>Field</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Versions</td>
<td>Shows all versions of the UI page. Use this list to compare versions or to revert to a previous version.</td>
</tr>
</tbody>
</table>

A UI page can be secured by creating an ACL with the following parameters:

- **Type**: `ui_page`
- **Operation**: `read`
- **Name**: name of the UI page to be protected

For details on creating an ACL rule, see [Create an ACL rule](#).

**UI page access**

Each UI page has a URL computed from the application scope, page name, and the `.do` file extension.

For example, to display the page called `glidewindow_example` on the demo system, you would navigate to `https://<instance name>.service-now.com/glidewindow_example.do`. If the page was part of a custom application called `example_app`, you would instead navigate to `https://<instance name>.service-now.com/x_example_app_glidewindow_example.do`.  

© 2020 ServiceNow, Inc. All rights reserved.  
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
You can also add additional parameters to a URL that can be accessed within a page's HTML section as jelly variables. That is, appending arguments to the URL as follows: /my_test_page.do?sysparm_verbose=true creates jelly variables called **verbose** that can be accessed as follows:

```java
<j2:if test="$[!empty(sysparm_verbose)]"> <span>show extra stuff </span> </j2:if>
```

A common practical example of this might be retrieving a database record for display. To build a list of a user's roles, pass in a parameter with the user's sys_id. Invoke the following UI page to display a list of roles for that user with Jelly code:

role_select.do?sysparm_user=5137153cc611227c000bbd1bd8cd2007

```java
<j:set var="jvar_user_id" value="$[sysparm_user]"/>

<g:evaluate>
var userRoles = new GlideRecord('sys_user_has_role');
userRoles.addQuery('user',"${jvar_user_id}");
userRoles.query();
</g:evaluate>

<select id='select_role'>
<j:while test="${userRoles.next()}"> 
  <option value="${userRoles.sys_id}"> ${userRoles.role.name} </option>
</j:while>
</select>
```

An exception to be careful of, though, is the reserved variable name sys_id. This variable always contains the ID of the UI page itself, regardless of what is specified in the URL. A common substitute variable name is sysparm_id.

Do not use URL parameters to load client scripts in UI pages. The system no longer evaluates scripts that are passed by URL parameter. If your implementation depends on this behavior, you can add a system property [glide.security.disable_ui_pages_sysparm_client_script] and set it to false to temporarily allow the evaluation of URL parameters passing scripts in UI pages.

**UI page process scripts**

If your UI page contains a form (uses the `<g:form>` tag), you can submit the form and have the process script run.

The processing script can naturally access fields on the form. For example, if your form contained the application_sys_id field:

```java
<g:ui_form><p>Click OK to run the processing script.</p>
  <g:dialog_buttons_ok_cancel ok = "return true"/> <input type = "hidden" name = "application_sys_id" value = "499836460a0a0b1700003e7ad950b5da"/> </g:ui_form>
```

You could then access it using just application_sys_id:

```java
var application = new GlideRecord('hr_application');
application.get(application_sys_id);
application.status = "Rejected";
application.update();
var urlOnStack = GlideSession.get().getStack().bottom();
```
response.sendRedirect(urlOnStack);

If you are using the UI page for a dialog, you can also reference the most recent URL on the stack using the code above and then send the response to that location. This is useful if you want to have the dialog’s processing script update something and then redisplay the screen that brought up the dialog.

UI Macros

UI macros are discrete scripted components administrators can add to the user interface.

UI macros are typically controls that provide inputs or information not provided by existing field types. By default, the system provides UI macros for a variety of user interface elements such as:

- All formatters
- The Service Catalog cart
- The action icons next to fields
- The action icons on forms and lists
- The widgets of a content management system
- The Orchestration activity designer

Administrators can create their own UI macros to provide custom controls or interfaces. Creating UI macros requires knowledge of Jelly script. Review the existing UI macros for examples and suggested approaches. Those who want to build custom interfaces with JavaScript technologies should consider Service Portal as an alternative.

Calling UI macros

Administrators can call UI macros from certain record types associated with the user interface.

**Calling UI Macros by record type**

<table>
<thead>
<tr>
<th>Record type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary attribute</td>
<td>Display an icon for a reference field:</td>
</tr>
<tr>
<td></td>
<td>ref_contributions=ui_macro_name</td>
</tr>
<tr>
<td>UI page</td>
<td>Display something on a UI page:</td>
</tr>
<tr>
<td></td>
<td>&lt;g:macro_invoke</td>
</tr>
<tr>
<td></td>
<td>macro=&quot;ui_macro_name&quot; /&gt;</td>
</tr>
<tr>
<td>UI macro</td>
<td>Call a UI macro from another UI macro:</td>
</tr>
<tr>
<td></td>
<td>&lt;?xml version= &quot;1.0&quot;</td>
</tr>
<tr>
<td></td>
<td>encoding=&quot;utf-8&quot; ?&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;j:jelly trim= &quot;false&quot;</td>
</tr>
<tr>
<td></td>
<td>xmlns:j=&quot;jelly:core&quot;</td>
</tr>
<tr>
<td></td>
<td>xmlns:g=&quot;glide&quot; xmlns:j2=&quot;null&quot;</td>
</tr>
<tr>
<td></td>
<td>xmlns:g2=&quot;null&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;g:ui_macro_name /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;g:ui_macro_name_2 /&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/j:jelly&gt;</td>
</tr>
</tbody>
</table>

**UI macro form**

Each UI macro record consists of a name and an XML document written in Jelly code.
UI macro fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A unique and descriptive name for this macro.</td>
</tr>
<tr>
<td>Active</td>
<td>Select the check box to render the element as defined. Clear the check box to disable the element without deleting the code. For example, the email_reply macro is inactive by default.</td>
</tr>
<tr>
<td>Description</td>
<td>Describe the purpose of the macro and parameters passed to it.</td>
</tr>
<tr>
<td>XML</td>
<td>Jelly script that defines the macro.</td>
</tr>
</tbody>
</table>

Processors

Processors provide a customizable URL endpoint that can execute arbitrary server-side JavaScript code and produce output such as TEXT, JSON, or HTML.

Typically, you create processors when you want a URL query to:

- Perform non-standard record operations.
- Contain complex logic as part of the API.
- Act on multiple tables.
- Create a logical API that can abstract implementation details.

When to create processors

Use this table to determine whether you should use existing functionality or create a processor.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform basic operations such as create, read, update, or delete records.</td>
<td>Use an existing web service, such as the REST API, or use an existing processor, such as Export to PDF.</td>
</tr>
<tr>
<td>Export data to a standard format such as JSON, CSV, PDF, Excel, or XML.</td>
<td></td>
</tr>
<tr>
<td>Operate on or export data from a single table.</td>
<td></td>
</tr>
<tr>
<td>Perform non-standard operations such as compute aggregation values.</td>
<td>Create a processor.</td>
</tr>
<tr>
<td>Export data in a non-standard format such as SQL.</td>
<td></td>
</tr>
<tr>
<td>Operate on or export data from multiple tables.</td>
<td></td>
</tr>
</tbody>
</table>

Processor form

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Unique name of the processor.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **Type** | Programming language of the processor script. Options include:  
- java: do not select this option  
- script |
| **Application** | Application containing this record. |
| **Active** | Flag to enable or disable the record. |
| **CSRF protect** | Option to protect the processor from running unless the instance uses a CSRF token. |
| **Description** | Description of the processor's function or purpose. |
| **Parameters** | List of available input parameters. Specify parameter values in the URL as `<parameter name>=<parameter value>`.  
**Note:** Parameter names must be processor-specific. Do not choose common parameter names that another processor might use. If you use a common parameter name, such as `id`, `sys_id`, or `table` in a processor, it can break other functionality, since the processor wins when that parameter exists in a URL. For example, a processor with an `id` parameter, regardless of the Path value in the same record, breaks the Service Portal, which depends on that parameter for page identification. |
| **Path** | URI path used to call this processor.  
Call a processor from the URL as:  
`https://<instance name>.service-now.com/<Path>.do` |
| **Script** | Immediately Invoked Function Expression to run when the system calls this processor.  
The function automatically provides input parameters for the following API objects.  
- `g_request`  
- `g_response`  
- `g_processor` |
### Protection policy
Policy to use to protect this record's script. Options include:
- None
- Read-only
- Protected

### Processor API components
Processors have access to dedicated API classes, objects, and methods.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g_response</td>
<td>An object of type HttpServletResponse. See GlideServletResponse.</td>
</tr>
<tr>
<td>setContentType('text/html; charset=UTF-8')</td>
<td>A GlideServletResponse method to set the content type of the response being sent to the client.</td>
</tr>
<tr>
<td>g_request</td>
<td>An object of type HttpServletRequest. See GlideServletRequest.</td>
</tr>
<tr>
<td>getParameter()</td>
<td>A glide method to get the value of a URL parameter.</td>
</tr>
<tr>
<td>canRead()</td>
<td>A GlideRecord method to determine if the user can read data from a table. See GlideRecord.</td>
</tr>
<tr>
<td>g_processor</td>
<td>A simplified servlet for processors. See GlideScriptedProcessor.</td>
</tr>
<tr>
<td>writeOutput()</td>
<td>A GlideScriptedProcessor method to display information on the client.</td>
</tr>
<tr>
<td>g_target</td>
<td>An object containing the target table name of a processor URL. For example, a processor containing the URI incident.do applies to the Incident table.</td>
</tr>
</tbody>
</table>

### Secure and protect a processor
You can protect your processor against unauthorized use by using role restrictions, and protect it by requiring a CSRF token.

You can re-use a table’s user role restrictions to protect it from access by your processor. This protection method assumes the processor will access table data.

1. Create or select a user role that has access to the table the processor script calls.
2. Navigate to **System Definition > Processors**.
3. In **Script**, add the following code block.

```javascript
var gr = new GlideRecord('your_table_name');
// canRead() compares the table’s ACL to the user making this request, and returns true if the logged-in user has read access to this table
if(gr.canRead())
{
  // Perform table query here
}
g_processor.writeOutput('Success!');
} else {
  g_processor.writeOutput('You do not have permission to read table your_table_name');
}

4. Update the code block to use other access restrictions as needed.

Available access functions include:
- canCreate()
- canRead()
- canWrite()
- canDelete()

5. Click Update.

Protect a processor with a CSRF token

You can protect a processor by requiring a CSRF token.

Script type processors can require a CSRF token check before the processor runs.

1. Navigate to System Definition > Processors.
2. Open a processor record.
3. Select the CSRF protect option.
4. Click Update.

Create a simple processor

Create a simple processor to execute a script from a URL query.

The following steps assume that you have your own demonstration instance.

1. Navigate to System Definition > Processors.
2. Click New.
3. Enter the following information.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Hello</td>
</tr>
<tr>
<td>Type</td>
<td>Script</td>
</tr>
<tr>
<td>Path</td>
<td>Hello</td>
</tr>
</tbody>
</table>
| Script  | var name= g_request.getParameter("name");
g_processor.writeOutput("text/plain","Hello "+name); |

4. Click Submit.
5. Enter a URL query to the instance with the following format: https://instance.service-now.com/processor_name.do?parameter=value. For example: https://<instancetype>.service-now.com/Hello.do?name=world.

Create a multi-table processor

Create a multi-table processor that reports the number of rows in any table on your instance.

The multi-table processor protects itself from performance and security violations by confirming that the user is authorized to read the table. It does not report on certain tables that are too large to query safely.

1. Navigate to System Definition > Processors.
The list of processors appears.

2. Click **New**.

3. Enter the following information.

<table>
<thead>
<tr>
<th>Name</th>
<th>TableSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Choose Javascript</td>
</tr>
<tr>
<td>Description</td>
<td>Return number of records in a table</td>
</tr>
<tr>
<td>Parameters</td>
<td>SIZE</td>
</tr>
<tr>
<td>Path</td>
<td>&lt;leave empty&gt;</td>
</tr>
</tbody>
</table>

**Script**

```javascript
var count = new GlideAggregate(g_target);
if( count.canRead() ) {
    count.addAggregate('COUNT');
    count.query();
    var records = 0;
    if (count.next()) {
        records = count.getAggregate('COUNT');
    }
    g_processor.writeOutput('table ' + g_target + ' has ' + records + ' records');
} else {
    g_processor.writeOutput('You do not have access to table ' + g_target);
}
```

4. Click **Save**.

5. Test the new processor by entering the following URLs:
   - `https://<instancename>.service-now.com/incident.do?SIZE`
   - `https://<instancename>.service-now.com/sys_user.do?SIZE`

Your instance reports the number of records in the table. For example, **table incident has 82 records**.

**Create a custom processor**

You can create a custom processor to execute a script from a URL query.

When complete, you will be able to:
- Create a new processor
- Run a script from a URL query

The following example steps assume you have your own demonstration instance.

1. Navigate to **System Definition > Processors**.
2. Click **New**.
3. For **Name**, enter **Hello**.
4. For **Type**, select **script**.
5. For **Path**, enter `Hello`.
6. For **Script**, enter the following code.

   ```javascript
   var name = g_request.getParameter("name");
   g_processor.writeOutput("text/plain", "Hello " + name);
   ```

7. Click **Submit**.
8. Logout of the instance and open a new browser window.
9. Enter a URL query to the instance with the following format: `https://instance.service-now.com/processor_name.do?parameter=value`. For example: `https://<instance name>.service-now.com/Hello.do?name=world`.
10. When prompted, enter credentials for valid user.

### Scripts - Background module

Administrators can use the Scripts - Background module to run arbitrary JavaScript code from the server.

The Scripts - Background module consists of the following components.

- A text field to enter JavaScript
- A selector to specify the application scope
- A **Run script** button
- A list of available scripts

Administrators can run any valid JavaScript that uses the **Glide API**. The system displays results, information, and error messages at the top of the screen.

**Note:** Running free-form JavaScript can cause system disruption or data loss. ServiceNow does not recommend running free-from scripts from a production instance.

By default, administrators can access this module without elevating privileges. If you want to require elevated privileges to access this module, set the system property `glide.script_processor.admin` to `security_admin`.

### Installation settings

Installation settings are global business rules with calculated names. Installation settings are calculated just before a record is displayed and facilitate dynamic determination of access and roles. Installation Settings permit the programmatic determination of a setting.

Installation settings controlling access to fields and records are:

- CanRead()
- CanWrite()
- CanCreate()
- CanDelete()

Functions can return true if access is permitted, false if not. No return value uses the permission calculated using roles. The function has access to the current record through the variable `current code`.

The name of the function checking the permission on a record is formed by prefixing the setting name with the record name:

- `record_nameCanRead()`
Similarly, the permission on a field in a record is formed by prefixing the function name with the record name, underscore, and field name:

```javascript
record_name_field_nameCanRead()
```

Naming examples:

```javascript
function incidentCanWrite() {} // can user write to this record?
function incident_numberCanWrite() {} // can user write to the number field?
```

This sample business rule restricts the writing of the name field in the `sys_dictionary` file when the entry exists:

```javascript
// the element name cannot be written unless this is a new record (not yet in database)
function sys_dictionary_nameCanWrite() {
    if (current.isNewRecord())
        return;
    return false;
}
```

### Using DurationCalculator to calculate a due date

Using the `DurationCalculator` script include, you can calculate a due date, using either a simple duration or a relative duration based on schedules.

The following script demonstrates how to use the global API to calculate a due date. The first part of the script illustrates how to set a start datetime using the method and then use the method to determine a due date that is ‘x’ amount of continuous time (seconds) from the specified start datetime. The second half of the script illustrates how to use `DurationCalculator` to calculate a due date based on a schedule. Schedules enable you to apply a “filter” on future time, such as only including the days in a work week within the calculation. For example, if you apply a schedule “weekdays” (which only includes Monday through Friday) to your duration calculation, and the start datetime is Friday at 5:00 pm, when you add a duration of two days, your due date would be Tuesday at 5:00 pm. If you did not use a schedule, your due date would be Sunday at 5:00 pm. For additional information on schedules, see [Schedules](#).

This script can be cut and pasted into the Scripts Background page and run as is. It can also serve as an example for authoring business rules, UI actions, or used any other place that server-side script can be authored.

```javascript
/**
 * Demonstrate the use of DurationCalculator to compute a due date.
 *
 * You must have a start date and a duration. Then you can compute a
 * due date using the constraints of a schedule.
 */
gs.include('DurationCalculator');
executeSample();

/**
 * Function to house the sample script.
 */
function executeSample(){
    // First we need a DurationCalculator object.
    var dc = new DurationCalculator();
```
// Simple computation of a due date without using a schedule. Seconds
// are added to the start date continuously to get to a due date.
var gdt = new GlideDateTime("2012-05-01 00:00:00");
dc.setStartDateTime(gdt);
if(!dc.calcDuration(2*24*3600)){ // 2 days
    gs.log("*** Error calculating duration");
    return;
}
gs.log("calcDuration no schedule: " + dc.getEndDateTime()); //
"2012-05-03 00:00:00" two days later

// Start in the middle of the night (2:00 am) and compute a due date 1
// hour in the future
// Without a schedule this yields 3:00 am.
var gdt = new GlideDateTime("2012-05-03 02:00:00");
dc.setStartDateTime(gdt);
if(!dc.calcDuration(3600)){
    gs.log("*** Error calculating duration");
    return;
}
gs.log("Middle of night + 1 hour (no schedule): " +
dc.getEndDateTime()); // No scheduled start date, just add 1 hour

// -------------- Add a schedule to the date calculator
---------------------
addSchedule(dc);

// Start in the middle of the night and compute a due date 1 hour in the
// future.
// Since we start at 2:00 am the computation adds the 1 hour from the
// start
// of the day, 8:00am to get to 9:00am
var gdt = new GlideDateTime("2012-05-03 02:00:00");
dc.setStartDateTime(gdt);
if(!dc.calcDuration(3600)){
    gs.log("*** Error calculating duration");
    return;
}
gs.log("Middle of night + 1 hour (with 8-5 schedule): " +
dc.getEndDateTime()); // 9:00 am

// Start in the afternoon and add hours beyond quitting time. Our
// schedule says the work day
// ends at 5:00pm, if the duration extends beyond that, we roll over to
// the next work day.
// In this example we are adding 4 hours to 3:00pm which gives us 10:00
// am the next day.
var gdt = new GlideDateTime("2012-05-03 15:00:00");
dc.setStartDateTime(gdt);
if(!dc.calcDuration(4*3600)){
    gs.log("*** Error calculating duration");
    return;
}
gs.log("Afternoon + 4 hour (with 8-5 schedule): " +
dc.getEndDateTime()); // 10:00 am.

// This is a demo of adding 2 hours repeatedly and examine the result.
This
// is a good way to visualize the result of a due date calculation.
var gdt = new GlideDateTime("2012-05-03 15:00:00");
dc.setStartDateTime(gdt);
for(var i=2; i<24; i+=1){
    if(!dc.calcDuration(i*3600)){
        gs.log("*** Error calculating duration");
        return;
    }
    gs.log("add "+ i +" hours gives due date: " + dc.getEndDateTime());
}

// Setting the timezone causes the schedule to be interpreted in the
// specified timezone.
// Run the same code as above with different timezone. Note that the 8
to 5 workday is
// offset by the two hours as specified in our timezone.
dc.setTimeZone("GMT-2");
var gdt = new GlideDateTime("2012-05-03 15:00:00");
dc.setStartDateTime(gdt);
for(var i=2; i<24; i+=1){
    if(!dc.calcDuration(i*3600)){
        gs.log("*** Error calculating duration");
        return;
    }
    gs.log("add "+ i +" hours gives due date (GMT-2): " +
            dc.getEndDateTime());
}
}

/**
 * Add a specific schedule to the DurationCalculator object.
 *
 * @param durationCalculator An instance of DurationCalculator
 */
function addSchedule(durationCalculator){
    // Load the "8-5 weekdays excluding holidays" schedule into our duration
    // calculator.
    var scheduleName = "8-5 weekdays excluding holidays";
    var grSched = new GlideRecord('cmn_schedule');
grSched.addQuery('name', scheduleName);
grSched.query();
    if(!grSched.next()){
        gs.log("*** Could not find schedule "+ scheduleName +""");
        return;
    }
    durationCalculator.setSchedule(grSched.getUniqueValue(),"GMT");
}

Using DurationCalculator to compute a simple duration
A simple duration is the number of seconds between two date times.
If no schedule is used then this is a simple time date subtraction. If a schedule is used, then the
schedule is consulted to remove non-work hours from the computation. Suppose schedule
"8-5 weekdays excluding holidays" is used. In this case, the number of work hours from noon
Monday to noon Tuesday is nine hours. To compute a simple duration, initialize the global API
DurationCalculator and call the method.

This script demonstrates how to use DurationCalculator to compute a simple duration.

/**
 * Sample script demonstrating use of DurationCalculator to compute simple
 * durations
 */
gs.include('DurationCalculator');
executeSample();
/**
 * Function to house the sample script.
 */
function executeSample(){
    // First we need a DurationCalculator object.
    var dc = new DurationCalculator();

    // Compute a simple duration without any schedule. The arguments
    // can also be of type GlideDateTime, such as fields from a GlideRecord.
    var dur = dc.calcScheduleDuration("2012-05-01", "2012-05-02");
    gs.log("calcScheduleDuration no schedule: " + dur); // 86400 seconds (24 hours)

    // The above sample is useful in limited cases. We almost always want
to
    // use some schedule in a duration computation, let's load a schedule.
    addSchedule(dc);

    // Compute a duration using the schedule. The schedule
    // specifies a nine hour work day. The output of this is 32400 seconds,
or
    // a nine hour span.
    dur = dc.calcScheduleDuration("2012-05-23 12:00:00","2012-05-24
12:00:00");
    gs.log("calcScheduleDuration with schedule: " + dur); // 32400 seconds
(9 hours)

    // Compute a duration that spans a weekend and holiday. Even though this
    // spans three days, it only spans 9 work hours based on the schedule.
    dur = dc.calcScheduleDuration("2012-05-25 12:00:00", "2012-05-29
12:00:00");
    gs.log("calcScheduleDuration with schedule spanning holiday: " + dur); //
32400 seconds (9 hours)

    // Use the current date time in a calculation. The output of this is
    // dependent on when you run it.
    var now = new Date();
    dur = dc.calcScheduleDuration("2012-05-15", new GlideDateTime());
    gs.log("calcScheduleDuration with schedule to now: " + dur); //
Different on every run.
}
/**
 * Add a specific schedule to the DurationCalculator object.
 *
 * @param durationCalculator An instance of DurationCalculator
 */
function addSchedule(durationCalculator){
    // Load the "8-5 weekdays excluding holidays" schedule into our duration
    // calculator.
    var scheduleName = "8-5 weekdays excluding holidays";
    var grSched = new GlideRecord('cmn_schedule');
grSched.addQuery('name', scheduleName);
grSched.query();
    if(!grSched.next()){
        gs.log('*** Could not find schedule "' + scheduleName + '"');
        return;
    }
    durationCalculator.setSchedule(grSched.getUniqueValue());
}
Using relative duration
Relative duration is very similar to simple duration except a piece of script is used to determine what parts of a day to remove from the difference calculation.

This script is stored in table cmn_relative_duration and can be examined by navigating to System Scheduler > Schedules > Relative Durations. There are some example relative duration scripts in the out-of-the-box instance.

A relative duration sys_id is passed to the method calcRelativeDuration() of the global API class after initialization. When this method is called, the DurationCalculator object is passed to the relative duration script (stored in table cmn_relative_duration) as the variable calculator. So, the relative duration script you write and store in cmn_relative_duration has access to the executing DurationCalculator through the variable calculator.

The following script demonstrates how to use DurationCalculator to calculate a relative duration.

```javascript
/**
 * Sample use of relative duration calculation.
 */
gs.include('DurationCalculator');
executeSample();

/**
 * Function to house the sample script.
 */
function executeSample(){
  // First we need a DurationCalculator object. We will also use
  // the out-of-box relative duration "2 bus days by 4pm"
  var dc = new DurationCalculator();
  var relDur = "3bf802c20a0a0b52008e2859cd8abcf2"; // 2 bus days by 4pm if
  // before 10am
  addSchedule(dc);

  // Since our start date is before 10:00am our result is two days from/
  // now at 4:00pm.
  var gdt = new GlideDateTime("2012-05-01 09:00:00");
  dc.setStartDateTime(gdt);
  if(!dc.calcRelativeDuration(relDur)){
    gs.log("*** calcRelativeDuration failed");
    return;
  }
  gs.log("Two days later 4:00pm: "+ dc.getEndDate());

  // Since our start date is after 10:00am our result is three days from
  // now at 4:00pm.
  var gdt = new GlideDateTime("2012-05-01 11:00:00");
  dc.setStartDateTime(gdt);
  if(!dc.calcRelativeDuration(relDur)){
    gs.log("*** calcRelativeDuration failed");
    return;
  }
  gs.log("Three days later 4:00pm: "+ dc.getEndDate());

  // Add a specific schedule to the DurationCalculator object.
  @param durationCalculator An instance of DurationCalculator
*/
function addSchedule(durationCalculator){
Using GlideRecord to query tables

In order to query a table, first create an object for the table.

This object is called a GlideRecord. To create a GlideRecord, create the following in script:

```javascript
var target = new GlideRecord('incident');
```

This creates a variable called target which is a GlideRecord object for the incident table. The only parameter needed is the name of the table to be accessed.

To process all records from the incident table, add the following script:

```javascript
target.query(); // Issue the query to the database to get all records
while (target.next()) {
    // add code here to process the incident record
}
```

This issues the `query()` to the database. Each call to `next()` would load the next record which you would process and do whatever you want to do.

But that is not the common case. Most of the time you actually want to retrieve a specific record or a specific set of records, and you have some criteria (query conditions) that define the records you want to obtain. For example, say you want to obtain all the incident records that have a priority value of 1. Here is the code that would accomplish that.

```javascript
var target = new GlideRecord('incident');
target.addQuery('priority',1);
target.query(); // Issue the query to the database to get relevant records
while (target.next()) {
    // add code here to process the incident record
}
```

Notice in the above code we added the line `target.addQuery('priority',1);`. This is indicating that you only want the records where the priority field is equal to 1. We assume that the majority of queries that you will want to do will be equality queries, queries where you want to find records where a field is equal to a value. Therefore we provide this format of the query and do not make you specify that you want an equals operation, we just assume it. However, lets say you wanted to find all incidents where the priority field is GREATER THAN 1. In that case you have to provide us with the operator that you want to apply to priority and this is done by providing the operator in the `addQuery()` request as is shown below.

```javascript
var target = new GlideRecord('incident');
target.addQuery('priority','>',1);
target.query(); // Issue the query to the database to get relevant records
while (target.next()) {
    // add code here to process the incident record
}
```
**Available JavaScript operators**
Describes the operators that can be used within an `addQuery()` request.

### Available JavaScript Operators

<table>
<thead>
<tr>
<th>Field</th>
<th>Definition</th>
<th><code>addQuery</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>=</code></td>
<td>Field must be equal to value supplied.</td>
<td><code>addQuery('priority', '=', 1);</code></td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Field must be greater than value supplied.</td>
<td><code>addQuery('priority', '&gt;', 1);</code></td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Field must be less than value supplied.</td>
<td><code>addQuery('priority', '&lt;', 3);</code></td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Field must be equal or greater than value supplied.</td>
<td><code>addQuery('priority', '&gt;=', 1);</code></td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Field must be equal or less than value supplied.</td>
<td><code>addQuery('priority', '&lt;=', 3);</code></td>
</tr>
<tr>
<td><code>!=</code></td>
<td>Field must not equal the value supplied.</td>
<td><code>addQuery('priority', '!=', 1);</code></td>
</tr>
<tr>
<td>STARTSWITH</td>
<td>Field must start with the value supplied. The example shown on the right returns all records where the <code>short_description</code> field starts with the text Error.</td>
<td><code>addQuery('short_description', 'STARTSWITH', 'Error');</code></td>
</tr>
<tr>
<td>CONTAINS</td>
<td>Field must contain the value supplied somewhere in the text. The example shown on the right returns all records where the <code>short_description</code> field contains the text Error anywhere in the field.</td>
<td><code>addQuery('short_description', 'CONTAINS', 'Error');</code></td>
</tr>
<tr>
<td>IN</td>
<td>Takes a map of values that allows commas, and gathers a collection of records that meet some other requirement. Behaves as <code>Select * from &lt;table&gt; where short_description IN ('Error', 'Success', 'Failure'),</code> which is identical to <code>Select * from &lt;table&gt; where short_description='Error'.</code> For example, to query all variable values that belong to a specific Activity, use the IN clause, and store their sys_ids in a map, or comma-separated list. Then query the variable value table and supply this list of sys_ids.</td>
<td><code>addQuery('short_description', 'IN', 'Error,Success,Failure');</code></td>
</tr>
</tbody>
</table>
### Field Definition addQuery

<table>
<thead>
<tr>
<th>Field</th>
<th>Definition</th>
<th>addQuery</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENDSWITH</td>
<td>Field must terminate with the value supplied. The example shown on the right returns all records where the short_description field ends with text Error.</td>
<td>addQuery('short_description', 'ENDSWITH', 'Error');</td>
</tr>
<tr>
<td>DOES NOT CONTAIN</td>
<td>Selects records that do NOT match the pattern in the field. This operator does not retrieve empty fields. For empty values, use the operators &quot;is empty&quot; or &quot;is not empty.&quot; The example shown on the right returns all records where the short_description field does not have the word &quot;Error.&quot;</td>
<td>addQuery('short_description', 'DOES NOT CONTAIN', 'Error');</td>
</tr>
<tr>
<td>NOT IN</td>
<td>Takes a map of values that allows commas, and gathers a collection of records that meet some other requirement. Behaves as: Select * from &lt;table&gt; where short_description NOT IN ('Error').</td>
<td>addQuery('short_description', 'NOT IN', 'Error', 'Success', 'Failure');</td>
</tr>
<tr>
<td></td>
<td>Special operator that retrieves only records of a specified &quot;class&quot; for extended tables. The code example on the right, shows how to retrieve all configuration items that are classified as computers.</td>
<td>addQuery('sys_class_name', 'INSTANCEOF', 'cmdb_ci_computer');</td>
</tr>
</tbody>
</table>

For additional information on the operators that are available for filters and queries, see Operators available for filters and queries.

There are also some special methods that you can use to search for data that is NULL or NOT NULL. To search for all incidents where the short_description field has not been supplied (is null), use the following query:

```javascript
var target = new GlideRecord('incident');
target.addNullQuery('short_description');
target.query(); // Issue the query to the database to get all records while (target.next()) {
    // add code here to process the incident record
}
```

To find all incidents in which a short_description has been supplied, use the following query:

```javascript
var target = new GlideRecord('incident');
target.addNotNullQuery('short_description');
target.query(); // Issue the query to the database to get all records while (target.next()) {
    // add code here to process the incident record
}
```
For more information on the GlideRecord API and its available methods, see .

GlideRecord query examples
These examples demonstrate how to perform various GlideRecord queries.

query

```javascript
var rec = new GlideRecord('incident');
rec.query();
while(rec.next()) {
    gs.print(rec.number + ' exists'); }
```

update

```javascript
var rec = new GlideRecord('incident');
rec.addQuery('active',true);
rec.query();
while(rec.next()) {
    rec.active = false;
    gs.print('Active incident ' + rec.number + ' closed');
    rec.update(); }
```

insert

```javascript
var rec = new GlideRecord('incident');
rec.initialize();
rec.short_description = 'Network problem';
rec.caller_id.setDisplayValue('Joe Employee');
rec.insert();
```

delete

```javascript
var rec = new GlideRecord('incident');
rec.addQuery('active',false);
rec.query();
while(rec.next()) {
    gs.print('Inactive incident ' + rec.number + ' deleted');
    rec.deleteRecord(); }
```

Querying Service Catalog Tables

You cannot directly query the variables of the Service Catalog Request Item table (sc_req_item). Instead, query the Variable Ownership table, (sc_item_option_mtom), by adding two queries, one for the variable name and another for the value. The query returns the many-to-many relationship, which you can dot-walk to the requested item. The following example finds the request items that have the variable item_name with a value of item_value and displays the request item numbers:

```javascript
var gr = new GlideRecord('sc_item_option_mtom');
gr.addQuery('sc_item_option.item_option_new.name','item_name');
gr.addQuery('sc_item_option.value','item_value');
```
gr.query();
while(gr.next()) {
    gs.addInfoMessage(gr.request_item.number); }

For additional information see .

**Running order guides automatically**

Service catalog order guides allow customers to make a single service catalog request that can generate several ordered items. Administrators can configure order guides to run automatically, from a workflow or a script to generate a set of ordered items without manually submitting a service catalog request. Administrators can also review and reprocess the order guide failures.

For example, an onboarding workflow for a new employee can run an order guide to automatically order items for that employee.

**Running order guides from scripts**

Running order guides with a server-side script is more complex than using workflows, but it allows more flexibility and can be used in non-workflow situations.

For example, you can use order guide scripts with UI actions or server-side business rules.

**Note:** When order guides run automatically, order guide UI policies are not enforced. Also, options in the Choose Options screen cannot be selected, so make sure that order guide rules define sensible defaults for these options to avoid processing failures.

Use the `SNC.ScriptableOrderGuide` Java class to run order guides with a script.

Use the `SNC.ScriptableOrderGuide(String orderGuideId)` constructor to create a new `ScriptableOrderGuide` object.
## Method summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>process(String json)</td>
<td>boolean</td>
<td>Runs the order guide using the JSON encoded string parameter as the input for the order guide. Returns true or false depending on whether processing was successful or not.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> Both opened_by and requested_for parameters must be passed to the order guide, and both must have valid user record sys_id values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If processing is successful and a request is created by the order guide, you can retrieve the request GlideRecord using getRequest().</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the processing fails, you can retrieve the failure GlideRecord using getFailure(), then submit the script for reprocessing using reprocess.</td>
</tr>
<tr>
<td>reprocess(GlideRecord failure)</td>
<td>boolean</td>
<td>Runs the order guide again using the JSON encoded string parameter stored in the failure GlideRecord.</td>
</tr>
<tr>
<td>getMessage()</td>
<td>String</td>
<td>Retrieves the message populated after processing or reprocessing.</td>
</tr>
<tr>
<td>getRequest()</td>
<td>GlideRecord</td>
<td>Retrieves the request GlideRecord.</td>
</tr>
<tr>
<td>getFailure()</td>
<td>GlideRecord</td>
<td>Retrieves the failure GlideRecord from the Scriptable Order Guide Failures (sc_script_order_guide_failure) table.</td>
</tr>
</tbody>
</table>

## Script example

This script processes an order guide called **IT Onboarding SOG**.

```java
// Creating the object to later be JSON encoded
```
var json = 
{"opened_by":"62826bf03710200044e0bfc8bcbe5df1","requested_for":"06826bf03710200044e0bfc8bcbe5d8a"};

var gr = new GlideRecord("sc_cat_item_guide");
if (gr.get("name","IT Onboarding SOG")) {
    var sog = new SNC.ScriptableOrderGuide(gr.getValue("sys_id"));
    var result = sog.process(new JSON().encode(json));
    if(!result)
        sog.getMessage();
    else {
        var request = sog.getRequest();
        gs.log("Request created - " + request.sys_id); 
    } 
}

Running order guides from workflows

Running an order guide from a workflow is suitable if you include order guides as part of a broader workflow-based process.

For example, an activity within an onboarding workflow for a new employee can automatically run an order guide to order items for that employee.

Note: When order guides run automatically, order guide UI policies are not enforced. Also, options in the Choose Options screen cannot be selected, so make sure that order guide rules define sensible defaults for these options to avoid processing failures.

To run order guides from a workflow, use the Scriptable Order Guide workflow activity.

Running order guides from workflows

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order Guide</td>
<td>The name of the order guide that this activity processes. For example, Example Employee Onboarding IT.</td>
</tr>
<tr>
<td>Script</td>
<td>A script passing information to the order guide. This information is sent as a JSON encoded string parameter assigned to the answer variable. The script must meet these requirements:</td>
</tr>
<tr>
<td></td>
<td>• The names of the variables in the script must match the names used within the order guide. For example, if the order guide uses a department variable in a rule condition, the script must also pass a department parameter.</td>
</tr>
<tr>
<td></td>
<td>• Both opened_by and requested_for parameters must be passed to the order guide, and both must have valid user record sys_id values.</td>
</tr>
</tbody>
</table>

Results

- **Success**: the activity successfully processed the order guide. This does not mean a request was created. If a request was created, the request sys_id is added to the workflow scratchpad under the sc_request variable.
• Failure: while processing the order guide a failure occurred, creating a failure record. If the processing fails, you can view and edit the failure record.

Workflow example

The Example Employee Onboarding IT Workflow workflow uses this example to generate IT catalog items for a new employee as part of an onboarding process.

The activity uses this script to:

1. Take a JSON string generated previously from the HR change record.
2. Append the mandatory opened_by and requested_for parameters to that string.
3. Submit the new string for processing by the order guide.

```
var parameters = new JSON().decode(current.payload);

// Need to amend the json object to include additional values.
parameters.opened_by = current.opened_by + "";
parameters.requested_for = current.opened_for + "";

answer = new JSON().encode(parameters);
```

View order guide failures

Order guide processing can fail, for example if the order guide being run does not exist. When a failure occurs during the order guide processing, the Scriptable Order Guide Failures submodule allows you to review and reprocess the failures. A record is created for each failure and once you fix the errors that caused the initial failure, you can reprocess the failed order guides.

If a failure occurs, a failure record is created in the Scriptable Order Guide Failures (sc_script_order_guide_failure) table.

To view details of a failure, navigate to Service Catalog > Catalog Administration > Scriptable Order Guide Failures, then open a failure record.

Reprocessing Failures

If you have fixed the error that caused the initial failure, you can reprocess failed order guides.

1. Navigate to Service Catalog > Catalog Administration > Scriptable Order Guide Failures.
2. Open the failure record.
3. Click the Reprocess related link.

To reprocess one or more failures:

1. Navigate to Service Catalog > Catalog Administration > Scriptable Order Guide Failures.
2. Select the check box beside one or more records to reprocess.
3. Select Reprocess from the Actions choice list.

Scriptable assignment of execution plans

Each catalog item has an associated execution plan, used whenever an item of that type is ordered; if no plan is specified, the default plan is used. This default is effective for most organizations, but your execution plan may need to vary based on additional criteria.
For example, in the base system service catalog, a request for a new PC always uses the PC Delivery Plan. However, this plan may need to vary for unusual circumstances - such as when a requester is working from home, at a remote location.

To provide this flexibility, you can use a script to override the default execution plan on a specific catalog item.

Limitations during script execution

Execution plan scripts have limitations that need to be considered during their implementation.

While the execution plan script runs:

- You cannot interact with any catalog tasks as catalog tasks are only created after the execution plan is selected.
- Some fields such as total delivery time and due date are not yet calculated, although the request itself is available within the script via current.request().
- Approvals have not yet been generated.

Writing the scripts

Follow these guidelines when writing execution plan scripts.

Execution plan scripts can access the same global variables and other functions as in any other server side execution plan.

- current is the currently-requested catalog item, sc_req_item.
- current.delivery_plan() is the assigned execution plan for this catalog item.

The evaluated value from the script is used as the sys_id of the execution plan.

Simple example:

```javascript
current.delivery_plan.setDisplayValue('PC Delivery Plan')
```

If an invalid value is returned, such as undefined or not found, then the existing assigned value is used.

More complex example:

```javascript
getexecutionplan();
function getexecutionplan() {
  var location = current.request.requested_for.location.getDisplayValue();
  // if we're in Atlanta
  if (location == 'Atlanta') {
    // use the remote pc delivery plan instead of the normal one
    var remote_plan = new GlideRecord('sc_cat_item_delivery_plan');
    remote_plan.addQuery('name', 'Remote PC Delivery Plan');
    remote_plan.query();
    remote_plan.next();
    current.delivery_plan = remote_plan.sys_id;
    return remote_plan_sys_id;
  }
  return current_delivery_plan;
}
```

In this example, any time a request is for a user in Atlanta, ServiceNow uses the Remote PC Delivery Plan. Otherwise, the execution plan is not overridden and ServiceNow uses the catalog item's normal execution plan, the PC Delivery Plan.

Add a script to a catalog item

You can add a script to a catalog item so that the script runs each time a user requests that item.

1. Navigate to Service Catalog > Maintain Items.
2. Select the relevant catalog item to which you wish to add the script.
3. Configure the catalog item form to add the execution plan script field, often named Delivery Plan Script.
4. Fill in the script details.
5. Update the item form with your changes.

The script runs each time that item is requested, selecting the execution plan to run with that item.
Use a script to approve an execution plan
You can use an approval rule script to approve an execution plan.

1. Retrieve an approval execution plan task.
2. View the Approval Script field.
3. Fill in an approval script using the same syntax and rules you would use on an approval rule.

For example, in the script below, the requester’s manager is the approver.
<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th>Manager Approval</th>
<th><strong>Upon approval</strong></th>
<th>Proceed to Next Task</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Order</strong></td>
<td>100</td>
<td><strong>Upon reject</strong></td>
<td>Cancel all future Tasks</td>
</tr>
<tr>
<td><strong>Delivery plan</strong></td>
<td>Ergonomic Furniture Delivery</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Delivery time</strong></td>
<td>Days: 2, Hours: 00:00:00</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SLA</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Condition</strong></td>
<td>Add Filter Condition, Add &quot;OR&quot; Clause</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Short description</strong></td>
<td>Doctor's recommendation required.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Instructions</strong></td>
<td>Verify that the employee provides a valid doctor's recommendation.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| **Approval script** | ```javascript
1 (function manager_as_approver () {
2 var request = current.request_item.request;
3 var rc = request.requested_for.manager;
4 return rc;
5 })();``` |                   |                      |
Scriptable service catalog variables

You can use scripting to reference any variables used by a request item from any table.

An example of a variable reference follows.

```
current.variables.<variable_name>
```

Where current refers to the current record, and `<variable_name>` is the name of your variable.

<i>Note: In order to reference a variable from JavaScript, it must have a name.</i>

Print a variable

```
var original = current.variables.original_number;
gs.print(original);
```

Set a variable

```
current.variables.name = "Auto-Generated:" + current.variables.asset_tag;
```

Create an inventory item with fields set from variables

```
doCreation();

function doCreation ( ) {
  var create = current.variables.create_item;
  if (create == 'true') { // we want to create an asset
    var computer = new GlideRecord('cmdb_ci_computer');
    computer.initialize();
    computer.asset_tag = current.variables.asset_tag;
    computer.serial_number = current.variables.serial_number;
    computer.name = current.variables.name;
    computer.manufacturer = current.variables.company;
    computer.insert();
  } }
```

Notes and limitations

1. You can only set a variable in a before business rule. Variables set in an after rule are not written to the database.
2. There is nothing in place to prevent namespace collision with variables. Creating two variables named computer_speed would result in only one of them showing up; the second one would overwrite the first one.
3. Date/time variables use the same time zone formatting and storage rules as all other dates in the system. They are stored internally in GMT, but translated into the user’s local time zone and format for display.
Service catalog script API

A scriptable API for the service catalog makes it easier to order from the catalog when using business rules.

The `Cart()` API allows you to order any quantity of catalog items, using the `sys_id` of the Catalog Item [sc_cat_item] you want. You can then set catalog variables to values as required, assuming the variables have names.

**Note:** If the script runs as a result of a scheduled import, the script runs as system or as the user specified by that import in the Run as field. The script uses the specified user’s cart. Each instantiation of a new `Cart()` object empties the cart of the calling user, but does not use or empty any other carts.

Examples

Ordering a single BlackBerry:

```javascript
var cartId = GlideGuid.generate(null);
var cart = new Cart(cartId);
var item = cart.addItem('e2132865c0a8016500108d9cee411699');
var rc = cart.placeOrder();
gs.addInfoMessage(rc.number);
```

Ordering twelve BlackBerries:

```javascript
var cartId = GlideGuid.generate(null);
var cart = new Cart(cartId);
var item = cart.addItem('e2132865c0a8016500108d9cee411699',12);
var rc = cart.placeOrder();
gs.addInfoMessage(rc.number);
```

Ordering an executive desktop and setting its OS:

```javascript
var cartId = GlideGuid.generate(null);
var cart = new Cart(cartId);
var item = cart.addItem('e46305bdc0a8010a00645e608031eb0f');
cart.setVariable(item,'os','Linux Red Hat');
var rc = cart.placeOrder();
gs.addInfoMessage(rc.number);
```

Setting a GlideRecord variable to null

GlideRecord variables (including current) are initially null in the database. Setting these back to an empty string, a space, or the JavaScript null value will not result in a return to this initial state.

**Note:** This functionality requires a knowledge of JavaScript.

**Note:** Functionality described here requires the Admin role.

To set it back to the initial state, simply set the value to “NULL”. Note that the `update()` function does not run on the current object but rather on the record. The object displays the initial value until it is called again from the record.
Example 1

```javascript
var gr1 = new GlideRecord('incident');
gr1.query();
while(gr1.next()) {
    gr1.priority = "NULL";
    gr1.update();
}
```

Example 2 (Business Rule)

```javascript
current.u_affected_value = 'NULL';
current.update();
```

Wizard scripts

You can create scripts to implement advanced custom functionality using wizards. Use wizard scripts in record generator panels, transitions, and UI policies.

ℹ️ **Note:** The System Wizards application is not active by default.

To reference a wizard variable in a script, use the format `wizard.<name>`, where `<name>` is the value in the **Name** field of the variable definition.

**Wizard script examples**

These examples demonstrate scripts that use wizard variables.

Example 1

```javascript
//Find a user in the sys_user table with the wizard variable 'EmpName'
function getUser(){
    var gu = new GlideRecord('sys_user');
    gu.addQuery('sys_id', wizard.EmpName);
    gu.query();
    if(gu.next()){
        gs.addInfoMessage('Found user ' + gu.name)
    }
}
```

Example 2

```javascript
//Loop through items in a wizard list collector with variable name astList
function removeAssets() {
    var items = wizard.astList.toString();
    items = items.split(',');
    for (var i =0; i < items.length; i++) {
        //got the asset id
        var sys_id = items[i];
    }
}
```

Example 3

```javascript
//Set the url on wizard completion - user will be redirected to this location
var uri = ('sys_user_list.do?sysparm_query=sys_id=' + wizard.EmpName);
wizard.redirect = uri;
```
Wizard scripts on record generators
Use wizard scripts in record generator panels to create records in any table, including on more than one table.

1. In the **Table** field, select **Global**.
2. Enter a script in the **Script** field. Configure the form to add the field, if necessary.

**Script:**
```javascript
var contract_id = createContract();
var uri = ('ast_contract.do?sysparm_query=sys_id=' + contract_id);
wizard.redirect = uri;
gs.addInfoMessage('Contract created');

function createContract(){
    var cc = new GlideRecord('ast_contract');
    //set values from wizard variables - note the format wizard.variable
    cc.sys_class_name = wizard.contract_type;
    cc.starts = wizard.starts;
    cc.ends = wizard.ends;
    cc.short_description = wizard.short_description;
    var cntr = cc.insert();
    associateAssets(cntr);
    return cntr;
}
```

Wizard UI policy and client scripts
Administrators and users with appropriate access rights can create dynamic effects for wizards using UI policies and client scripts, including:

- Get or set variable values
- Hide or display variables
- Make variables mandatory
- Validate form submission
UI policies apply effects based on conditions constructed with a condition builder. Client scripts accomplish more advanced functionality. Because UI policies do not require scripting, they are less likely to need maintenance after system updates.

**Note:** The System Wizards application is not active by default.

### Context-sensitive help

The syntax editor can display context-sensitive API information. Context-sensitive help includes the ability to:

- List script elements that are valid at the cursor's location. The system displays suggestions in a pop-up window.
- Add a selected script element at the cursor's location. If the cursor is within or adjacent to a partial entry, the system completes the entry with the selected script element.
- View API documentation for a selected suggestion.
- View the expected parameters and format of the current script element.

If the cursor is adjacent to a text string, the system searches for script elements that start with this text string. For example, while the cursor is within or adjacent to the string GlideR, the system displays script elements such as:

- GlideRecord
- GlideRecordSecure

Context-sensitive suggestions are based on script type. For example, when working on a business rule, only suggestions from the server API and for objects such as current and previous display. When working on a client script, the system only displays suggestions from the client API.

### Client scripts

Client scripts allow the system to run JavaScript on the client (web browser) when client-based events occur such as when a form loads, after form submission, or when a field changes value. Use client scripts to configure forms, form fields, and field values while the user is using the form. Client scripts can:

- make fields hidden or visible
- make fields read only or writable
- make fields optional or mandatory based on the user's role
- set the value in one field based on the value in other fields
- modify the options in a choice list based on a user's role
- display messages based on a value in a field

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the client script.</td>
</tr>
<tr>
<td>Table</td>
<td>Table to which the client script applies.</td>
</tr>
<tr>
<td>UI Type</td>
<td>Target user interface to which the client script applies.</td>
</tr>
</tbody>
</table>
### Type

- **onLoad()** — runs when the system first renders the form and before users can enter data. Typically, `onLoad()` client scripts perform client-side manipulation of the current form or set default record values.

- **onSubmit()** — runs when a form is submitted. Typically, `onSubmit()` scripts validate things on the form and ensure that the submission makes sense. An `onSubmit()` client script can cancel form submission by returning a value of false.

- **onChange()** — runs when a particular field value changes on the form. The `onChange()` client script must specify these parameters.
  - control: the DHTML widget whose value changed.
  - oldValue: the value the widget had when the record was loaded.
  - newValue: the value the widget has after the change.
  - isLoading: identifies whether the change occurs as part of a form load.
  - isTemplate: identifies whether the change occurs as part of a template load.

- **onCellEdit()** — runs when the list editor changes a cell value. The `onCellEdit()` client script must specify these parameters.
  - sysIDs: an array of the sys_ids for all items being edited.
  - table: the table of the items being edited.
  - oldValues: the old values of the cells being edited.
  - newValue: the new value for the cells being edited.
  - callback: a callback that continues the execution of any other related cell edit scripts. If `true` is passed as a parameter, the other scripts are executed or the change is committed if there are no more scripts. If `false` is passed as a parameter, any further scripts are not executed and the change is not committed.

**Note:** `onCellEdit()` client scripts do not apply to dashboard list widgets.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Application where this client script resides.</td>
</tr>
<tr>
<td>Active</td>
<td>Enables the client script when selected. Unselect this field to disable the client script.</td>
</tr>
<tr>
<td>Inherited</td>
<td>Indicates whether the client script applies to extended tables.</td>
</tr>
<tr>
<td>Global</td>
<td>If true, the client script runs on all views of the table.</td>
</tr>
<tr>
<td>View</td>
<td>Only visible when Global is unselected. Views on which the client script will run.</td>
</tr>
<tr>
<td>Description</td>
<td>Content describing the functionality and purpose of the client script.</td>
</tr>
<tr>
<td>Messages</td>
<td>Text string (one per line) available to the client script as localized messages using <code>getmessage('[message]')</code>. For additional information, see Translate a client script message.</td>
</tr>
<tr>
<td>Script</td>
<td>Contains the client script.</td>
</tr>
</tbody>
</table>
Where client scripts run

With the exception of `onCellEdit()` client scripts, client scripts only apply to forms and search pages. If you create a client script to control field values on a form, you must use one of these other methods to control field values when on a list.

- Create an access control to restrict who can edit field values.
- Create a business rule to validate content.
- Create a data policy to validate content.
- Create an `onCellEdit()` client script to validate content.
- Disable list editing for the table.

Client API

The client-side Glide API provides classes and methods that you can use in client scripts.

UI scripts

UI scripts provide a way to package client-side JavaScript into a reusable form, similar to how script includes store server-side JavaScript. Administrators can create UI scripts and run them from client scripts and other client-side script objects and from HTML code.

UI scripts are not supported for mobile.

**Global UI scripts**

You can create a UI script and designate it as global, which makes the script available on any form in the system. You cannot create a global UI script in a scoped application.

You can mark a UI script as Global to make it available on any form in the system. For example, you can create a UI script that has a function `helloWorld()`, and has the **Global** field checked:

```javascript
function helloWorld() {
    alert('Hi');
}
```

After you create this global UI script, you can call the `helloWorld()` function from any client script or UI policy you write.

**Create a UI script**

Create a UI script to define reusable client-side JavaScript code.

To create UI scripts, navigate to **System UI > UI Scripts** and create or edit a record (see table for field descriptions).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script Name</td>
<td>Name of the UI script. Ensure the name is unique on your system.</td>
</tr>
<tr>
<td>API Name</td>
<td>The API name of the UI script, including the scope and script name (for example, x_custom_app.HelloWorld).</td>
</tr>
<tr>
<td>Application</td>
<td>Application that contains the UI script.</td>
</tr>
<tr>
<td>Active</td>
<td>Indicator of whether the UI script is active. Only active UI scripts can run.</td>
</tr>
</tbody>
</table>
### Field	|	Description
---|---
Global	|	Indicator of whether the script loads on every page in the system.

**Note:** Use caution when creating global UI scripts because they can impact performance. You cannot create a global UI script in a scoped application.

**Description**
Summary of the purpose of the script.

**Script**
Client-side script to run when called from other scripts.

---

**Run UI scripts**
Follow these guidelines when running UI scripts.

To run a UI script on a form, create a formatter. In the associated UI macro, include a `g:requires` tag and specify the `name` parameter as the name of the UI script followed by the `.jsdbx` extension. Add the formatter on the form view.

This code ensures that the definitions and results of the UI script are immediately available in the browser.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<j:jelly trim="false" xmlns:j="jelly:core" xmlns:g="glide" xmlns:j2="null"
 xmlns:g2="null">
  <g2:evaluate var="jvar_stamp">
    var gr = new GlideRecord('sys_ui_script');
    gr.orderByDesc('sys_updated_on');
    gr.query();
    gr.next();
    gr.getValue('sys_updated_on');
  </g2:evaluate>
  <g:requires name="<UI SCRIPT NAME>.jsdbx" params="cache=$[jvar_stamp]" />
</j:jelly>
```

To run a UI script from HTML code, use the `<script>` tag and specify the `src` argument as the API name of the UI script followed by the `.jsdbx` extension. For example, include the UI script named **CoolClock** with this code:

```html
<script language="javascript" src="CoolClock.jsdbx" />
```

---

**Client script design and processing**

Well-designed client scripts can reduce the amount of time it takes users to complete a form.

Proper client-side processing depends on the form loading first. Making record updates prior to form load can produce unexpected results that bypass client-side processing.

If you create client scripts to control field values on a form, you must use another method to control these field values in a list. You can:

- Disable list editing for the table.
- Create appropriate business rules or access controls for list editing.
- Create data policies.
- Create a separate onCellEdit client script.

**Restrict list editing**
If you create UI policies or client scripts for fields on a form, you must use another method to ensure that data in those fields is similarly controlled in a list.

With the exception of onCellEdit client scripts, UI policies and client scripts apply to forms only. Use the following methods to restrict list editing when using client scripts:

- Disable list editing for the table.
- Create appropriate business rules or access controls for list editing.
- Create data policies.
- Create a separate onCellEdit client script.

**Minimize server lookups**
Use client data as much as possible to eliminate the need for time-consuming server lookups.

Client scripting uses either data available on the client or data retrieved from the server. The top ways to get information from the server are g_scratchpad and asynchronous GlideAjax lookup.

The primary difference between these methods is that g_scratchpad is sent once when a form is loaded (information is pushed from the server to the client), whereas GlideAjax is dynamically triggered when the client requests information from the server.

**Note:** GlideRecord and g_form.getReference() are also available for retrieving server information. However, these methods are no longer recommended due to their performance impact. Both methods retrieve all fields in the requested GlideRecord when most cases only require one field.

**Retrieve server data using g_scratchpad**
The g_scratchpad object passes information from the server to the client, such as when the client requires information not available on the form.

For example, if you have a client script that needs to access the field u_retrieve, and the field is not on the form, the data is not available to the client script. A typical solution to this situation is to place the field on the form and then always hide it with a client script or UI policy. While this solution may be faster to configure, it is slower to execute.

If you know what information the client needs from the server before the form is loaded, a display business rule can create g_scratchpad properties to hold this information. The g_scratchpad is sent to the client when the form is requested, making it available to all client-side scripting methods. This is a very efficient means of sending information from the server to the client. However, you can only load data this way when the form is loaded. The business rule cannot be triggered dynamically. In those cases, use an asynchronous GlideAjax call.

For example, assume you open an incident and need to pass this information to the client:

- The value of the system property css.base.color
- Whether or not the current record has attachments
- The name of the caller’s manager

A display business rule sends this information to the client using the following script:

```javascript
g_scratchpad.css = gs.getProperty('css.base.color');
g_scratchpad.hasAttachments = current.hasAttachments();
```
g_scratchpad.managerName =
current.caller_id.manager.getDisplayValue();

To access scratchpad data using a client script:

```javascript
// Check if the form has attachments
if (g_scratchpad.hasAttachments)
  // do something interesting here
else
  alert('You need to attach a form signed by ' +
g_scratchpad.managerName);
```

**Retrieve server data using asynchronous GlideAjax**

Asynchronous GlideAjax allows you to dynamically request information from the server.

This script compares the support group of the CI and the assignment group of the incident by name:

```javascript
// Alert if the assignment groups name matches the support group
function onChange(control, oldValue, newValue, isLoading) {
  if (isLoading)
    return;

  var ga = new GlideAjax('ciCheck');
  ga.addParam('sysparm_name', 'getCiSupportGroup');
  ga.addParam('sysparm_ci', g_form.getValue('cmdb_ci'));
  ga.addParam('sysparm_ag',
g_form.getValue('assignment_group'));
  ga.getXML(doAlert); // Always try to use asynchronous
                      // (getXML) calls rather than synchronous (getXMLWait)
}

function doAlert(response) {
  var answer =
    response.responseXML.documentElement.getAttribute("answer");
  alert(answer);
}
```

This script relies on the accompanying script include:

```javascript
var ciCheck = Class.create();

ciCheck.prototype = Object.extendObject(AbstractAjaxProcessor, {
  getCiSupportGroup: function() {
    var retVal = ''; // Return value
    var ciID   = this.getParameter('sysparm_ci');
    var agID   = this.getParameter('sysparm_ag');
    var ciRec  = new GlideRecord('cmdb_ci');

    // If we can read the record, check if the sys_ids match
```
if (ciRec.get(ciID)) {
  if (ciRec.getValue('support_group') == agID)
    retVal = 'CI support group and assignment group match';
  else
    retVal = 'CI support group and assignment group do not match';
  // Can't read the CI, then they don't match
} else {
  retVal = 'CI support group and assignment group do not match';
}
return retVal;
}};

**Use the setValue() displayValue parameter for reference fields**

When using setValue() on a reference field, include the displayValue parameter to avoid additional server calls.

When using `setValue()` on a reference field, be sure to include the reference field display value as the 3rd parameter. If you set the value without the displayValue, the instance does a synchronous call to retrieve the display value for the record you specified. This extra round trip to the server can impact performance.

This example demonstrates the incorrect way to call `setValue`:

```javascript
var id = '5137153cc611227c000bbd1bd8cd2005';

var name = 'Fred Luddy';
g_form.setValue('assigned_to', id, name); // No server call required
```

```
Use UI policy instead of a client script
When possible, consider using a UI policy instead of a client script.

UI policies provide these benefits over client scripts:

- UI policies have an Order field to allow full control over the order in which client-side operations take place.
- UI policies do not require scripting to make a field mandatory, read-only, or visible.
```
Validate input using client scripts
An excellent use for client scripts is validating input from the user.
This validation improves the user experience because the user finds out if there are data issues before submitting the information.

An example of validation is to verify that the Impact field value is valid with the Priority field value. In this example, Low impact is not allowed with High priority.

```javascript
if (g_form.getValue('impact') == '3' &&
    g_form.getValue('priority') == '1')
    g_form.showFieldMsg('impact', getMessage('Low impact now allowed with High priority'), 'error');
```

Set client script order
Control the order of execution for your client scripts using the Order field. To avoid having two or more client scripts run concurrently and then conflict, you can add an order for the scripts to run in.

Role required: admin
Adding an order to the client script creates a processing sequence, ordered from lowest to highest number. If two scripts conflict, the client script with the lower number executes first.

1. Navigate to System Definition > Client Script and open an existing client script or click New.
2. Configure the form to include the Order field.
3. Add a number to the order field based on what order you want it to run in relation to other client scripts. Choose a lower number for the script you want to execute first.

Avoid DOM manipulation
Avoid Document Object Model (DOM) manipulation if possible. It can cause a maintainability issue when browsers are updated.
Instead, use the GlideForm API or consider a different approach for the solution. In general, when using DOM manipulation methods, you have to reference an element in the DOM by ID or using a CSS selector. When referencing out-of-box DOM elements, there is a risk that the element ID or placement within the DOM could change, causing the code to stop working and/or generate errors. Use forethought, caution, and have a full understanding of the risk you are incurring. Review these objects and reduce the use of DOM manipulation methods as much as possible.

Avoid global client scripts
A global client script is any client script where the selected Table is Global. Global client scripts have no table restrictions, therefore they will load on every page in the system introducing browser load delay in the process.
There is no benefit to loading this kind of scripts on every page.
As an alternative, and for a more modular and scalable approach, consider moving client scripts to a base table (such as Task(task) or Configuration Item(cmdb_ci)) that can be inherited for all the child/extending tables. This eliminates the system loading the scripts on every form in the UI - such as home pages or Service Catalog where they are rarely (if ever) needed.

Enclose code in functions
Enclose the code in a client script inside a function.

Client scripts without a function cause issues with variable scope. When code is not enclosed in a function, variables and other objects are available and shared to all other client-side scripts.
If you are using the same variable names, it is possible that they could collide. This can lead to unexpected consequences that are difficult to troubleshoot.

Consider this example:

```javascript
var state = "6";

function onSubmit() {
    if(g_form.getValue('incident_state') == state) {
        alert("This incident is Resolved");
    }
}
```

Because the `state` variable is not enclosed in a function, all client-side scripts, have access to it. Other scripts may also use the common variable name `state`. The duplicate names can conflict and lead to unexpected results. These issues are difficult to isolate and resolve. To avoid this issue, ensure that all code is wrapped in a function:

```javascript
function onSubmit() {
    var state = "6";
    if(g_form.getValue('incident_state') == state) {
        alert("This incident is Resolved");
    }
}
```

This solution is much safer because the scope of the variable `state` is limited to the `onSubmit()` function. Therefore, the `state` variable does not conflict with `state` variables in other client-side scripts.

**Run only necessary scripts**

To avoid running time-consuming scripts unnecessarily, make sure that client scripts perform only necessary tasks.

The following examples demonstrate improvements to the initial code sample. Each example demonstrates a particular enhancement to the script to improve performance and avoid unnecessary calls.

Remember that client scripts have no **Condition** field. This means that `onLoad()` and `onChange()` scripts run in their entirety every time the appropriate form is loaded. This example is an inefficient `onChange()` client script set to run when the **Configuration item** field changes.

```javascript
//Set Assignment Group to CI's support group if assignment group is empty
function onChange(control, oldValue, newValue, isLoading) {
    var ciSupportGroup = g_form.getReference('cmdb_ci').support_group;
    if (ciSupportGroup == '' && g_form.getValue('assignment_group') == '')
        g_form.setValue('assignment_group', ciRec.support_group.sys_id);
}
```
This example improves upon the first by replacing the getReference() or GlideRecord lookup with an asynchronous GlideAjax call.

```javascript
//Set Assignment Group to support group if assignment group is empty
function onChange(control, oldValue, newValue, isLoading) {
    var ga = new GlideAjax('ciCheck');
    ga.addParam('sysparm_name', 'getSupportGroup');
    ga.addParam('sysparm_ci', g_form.getValue('cmdb_ci'));
    ga.getXML(setAssignmentGroup);
}

function setAssignmentGroup(response) {
    var answer = response.responseXML.documentElement.getAttribute("answer");
    g_form.setValue('assignment_group', answer);
}
```

The `isLoading` flag is the simplest way to prevent unnecessary code from taking up browser time in onChange scripts. The `isLoading` flag should be used at the beginning of any script that is not required to run when the form is loading. There is no need to run this script on a form load because the logic would have already run when the field was last changed. Adding the `isLoading` check to the script prevents it from doing a `cmdb_ci` lookup on every form load.

The `isTemplate` flag indicates that a template is loading.

```javascript
//Set Assignment Group to CI's support group if assignment group is empty
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
    if (isLoading)
        return;
    var ga = new GlideAjax('ciCheck');
    ga.addParam('sysparm_name', 'getSupportGroup');
    ga.addParam('sysparm_ci', g_form.getValue('cmdb_ci'));
    ga.getXML(setAssignmentGroup);
}

function setAssignmentGroup(response) {
    var answer = response.responseXML.documentElement.getAttribute("answer");
    g_form.setValue('assignment_group', answer);
}
```

If the onChange script should run during loading, use the following convention:

```javascript
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
```
if (isLoading) {}; // run during loading
    // rest of script here
}

The `newValue` check tells this script to continue only if there is a valid value in the relevant field. This prevents the script from running when the field value is removed or blanked out. This also ensures that there will always be a valid value available when the rest of the script runs.

```
//Set Assignment Group to CI's support group if assignment group is empty
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
    if (isLoading)
        return;

    if (newValue) {
        var ga = new GlideAjax('ciCheck');
        ga.addParam('sysparm_name', 'getSupportGroup');
        ga.addParam('sysparm_ci', g_form.getValue('cmdb_ci'));
        ga.getXML(setAssignmentGroup);
    }
}

function setAssignmentGroup(response) {
    var answer = response.responseXML.documentElement.getAttribute("answer");
    g_form.setValue('assignment_group', answer);
}
```

To have the script react to a value that changes after the form loads, use the `newValue != oldValue` check.

**Note:** This example does not catch users changing a value and then changing it back to its original value.

```
//Set Assignment Group to CI's support group if assignment group is empty
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
    if (isLoading)
        return;

    if (newValue) {
        if (newValue != oldValue) {
            var ga = new GlideAjax('ciCheck');
            ga.addParam('sysparm_name', 'getSupportGroup');
        }
    }
```
ga.addParam('sysparm_ci',
g_form.getValue('cmdb_ci'));
ga.getXML(setAssignmentGroup);
}
}
function setAssignmentGroup(response) {
    var answer =
        response.responseXML.documentElement.getAttribute("answer");
    g_form.setValue('assignment_group', answer);
}

In this example, the GlideAjax call is buried one level deeper by rearranging the script to check as many things available to the client as possible before running the server calls. The script checks the assignment before executing the GlideAjax call. This prevents the server lookup when the `assignment_group` field is already set.

//Set Assignment Group to CI's support group if assignment group is empty
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
    if (isLoading)
        return;

    if (newValue) {
        if (newValue != oldValue) {
            if (g_form.getValue('assignment_group') == '') {
                var ga = new GlideAjax('ciCheck');
                ga.addParam('sysparm_name', 'getSupportGroup');
                ga.addParam('sysparm_ci',
g_form.getValue('cmdb_ci'));
                ga.getXML(setAssignmentGroup);
            }
        }
    }
}

function setAssignmentGroup(response) {
    var answer =
        response.responseXML.documentElement.getAttribute("answer");
    g_form.setValue('assignment_group', answer);
}

Catalog client script creation

Client-side scripts can add dynamic effects and validation to forms. Scripts can apply to service catalog items or variable sets, allowing administrators to use the same functionality that is available on other forms.

You can use client side scripts to:
- Get or set variable values.
• Hide or display variables.
• Make variables mandatory or not.
• Validate form submission.
• Add something to the cart.
• Order something immediately.

Catalog client script considerations
When you create catalog client scripts, be aware of the following considerations.

• Catalog client scripts run when a user orders an item from the service catalog. Catalog client scripts can also run when variables or variable sets for a catalog item are displayed when a user requests that item.
• For a variable to be accessible using a catalog client script, it must have a variable name. Variables without names do not appear in the list of available variables.
• When using standard client scripts on a Requested Item or Catalog Task form, make a note of fields with the same name as variables. If a table field and a variable of the same name are both present on a form, the table field is matched when it is accessed using a script. If this happens, specifically address the variable by naming it `variables.variable_name`. For example:
  ```javascript
  g_form.setValue('variables.replacement', 'false');
  ```
• If you are using record producers to pass variables from the service catalog to other types of records, these variables are made visible in those records with a variable editor, such as the Change Variable Editor UI formatter on Change request forms. You can manipulate these variables using standard client script methods, such as `setDisplay`, `setMandatory`, `setValue`, and `getValue`.
• Catalog client scripts can be used for catalog items included in a wizard.
• You can use the `g_form.refreshSlushbucket(fieldName)` API to update a list collector variable.

Catalog client script differences
Catalog client scripts are very similar to standard client scripts, with a few important differences.

• Instead of selecting a table such as Incident for the script, select a catalog item or variable set. As your system may have a large number of catalog items, you should select a catalog item or variable set using a reference field instead of the choice list that the standard Client Script form uses.
• When using an `onChange()` catalog client script, it is linked to a particular variable instead of a field. The system automatically populates the variable name selection list with any named variables from the catalog item or variable set selected.

Create a catalog client script
Follow this procedure to create a catalog client script.

1. Navigate to Service Catalog > Catalog Administration > Catalog Client Scripts. A list of current custom catalog client scripts appears.
2. Click New.
3. Fill in the fields, as appropriate (see table).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a unique name for the catalog client script.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Applies to</td>
<td>Select the item type this client script applies to:</td>
</tr>
<tr>
<td></td>
<td>• A Catalog Item: enables the Catalog item field.</td>
</tr>
<tr>
<td></td>
<td>• A Variable Set: enables the Variable set field.</td>
</tr>
<tr>
<td>Active</td>
<td>Select the check box to enable the client script. Clear the check box to disable the script.</td>
</tr>
<tr>
<td>UI Type</td>
<td>Whether to apply this to desktop, mobile, or both.</td>
</tr>
<tr>
<td>Script</td>
<td>Enter the client script that should run on the service catalog item.</td>
</tr>
<tr>
<td>Type</td>
<td>Select when the script should run, such as <code>onLoad</code> or <code>onSubmit</code>.</td>
</tr>
<tr>
<td>Catalog item or Variable set</td>
<td>Select a catalog item or variable set from the list. The field name and options available depend on the selection in the Applies to field.</td>
</tr>
<tr>
<td>Applies on a Catalog Item view</td>
<td>Select the check box to apply the catalog client script to catalog items displayed within the order screen on the service catalog. Available in the requester view.</td>
</tr>
<tr>
<td>Applies on Requested Items</td>
<td>Select the check box to apply the catalog client script on a Requested Item form, after the item is requested. Available in the fulfiller view. See VEditor.</td>
</tr>
<tr>
<td>Applies on Catalog Tasks</td>
<td>Select the check box to apply the catalog client script when a Catalog Task form for the item is being displayed. Available in the fulfiller view. See VEditor.</td>
</tr>
<tr>
<td>Applies on the Target Record</td>
<td>Select the check box to support the catalog UI policy on a record created for task-extended tables via record producers. See Default variable editor.</td>
</tr>
</tbody>
</table>

4. Click **Submit**.

*Catalog client script examples*

Examples of client scripts to perform common actions.

**Get the value of a variable**

Use the following syntax to obtain the value of a catalog variable. Note that the variable must have a name. Replace `variable_name` with the name of the variable.

```javascript
g_form.getValue('variable_name');
```
Restrict the number of characters a user can type in a variable

This is an example of a script that runs when the variable is displayed, rather than when the item is ordered.

```javascript
function onLoad(){
    var sd = g_form.getControl('short_description');
    sd.maxLength=80;
}
```

Mobile client GlideForm (g form) scripting and migration

Client scripting for mobile is identical to scripting for the web, with some exceptions. All new scripts must conform to certain guidelines. The following items are affected on the mobile platform: client scripts, UI policies, navigator modules, and UI actions.

Client scripts

For new or existing scripts to be valid for mobile, they must conform to the following requirements:

- Use the new mobile methods in place of `g_form.getControl()`.
- Do not use deprecated methods.
- Do not reference unsupported browser objects.
- Do not make synchronous JavaScript, GlideAjax, and GlideRecord calls.
- Do not call methods that are not available for mobile.
- Enable scripts to run on the mobile UI.

Requirements

| Use the new mobile methods | Several new methods are available for modifying form fields instead of directly manipulating the HTML. These methods replace previous usages of `g_form.getControl()`, which is deprecated for the mobile platform. In your existing scripts, ensure that the new methods are used in place of methods that are not valid on the mobile platform. For information on these new methods, refer to . |
| Do not use deprecated methods | The following methods have been deprecated for the mobile platform because direct access to HTML elements is not allowed:

- `g_form.getControl()`
- `g_form.getFormElement()`
- `g_form.getElement()`

To ensure that existing scripts are compatible, remove all calls to deprecated methods from your code. For new scripts, do not use deprecated methods if you want the script to be valid for mobile.

For `g_form.getControl()`, some of the functionality previously included with this method has been extracted to individual methods. Instead of `g_form.getControl()`, use the new methods described on the ServiceNow® Developer Site. |
| Do not reference unsupported browser objects | The following browser objects are not supported in mobile scripts:

- Window
- jQuery or Prototype ($, $j, or $$)
- Document

Make sure that new scripts do not use these objects, and remove any usage of these objects from your existing scripts. Use GlideForm (g_form) instead, which provides methods such as `setLabel()`, `addDecoration()`, and `hasField()` for accomplishing the same tasks. |
<table>
<thead>
<tr>
<th>Do not make synchronous JavaScript calls</th>
<th>The mobile platform does not allow synchronous JavaScript calls. The <code>g_form.getReference()</code> method must now have the callback parameter defined. For example:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>g_form.getReference(fieldName, callback)</code></td>
</tr>
<tr>
<td></td>
<td>Be sure that all <code>g_form.getReference()</code> calls include the callback parameter. For example, the following script does not include a callback and is incompatible with the mobile platform:</td>
</tr>
<tr>
<td></td>
<td><code>var userName = g_form.getReference('assigned_to').user_name, g_form.setValue('u_assigned_user_name', userName);</code></td>
</tr>
<tr>
<td></td>
<td>The following script has been updated to include the callback and is compatible with the mobile platform:</td>
</tr>
<tr>
<td></td>
<td><code>g_form.getReference('assigned_to', function(gr) { g_form.setValue('u_assigned_user_name', gr.user_name); });</code></td>
</tr>
</tbody>
</table>

| Do not make synchronous Ajax calls     | The mobile platform does not allow synchronous GlideAjax calls. Any use of `getXMLWait()` in a GlideAjax call will not work on the mobile platform. Be sure that all GlideAjax calls are asynchronous. For more on synchronous versus asynchronous GlideAjax calls and `getXMLWait()`, see AJAX. For information on the available GlideAjax methods, refer to the. |
| Do not make synchronous GlideRecord calls | The mobile platform does not allow synchronous calls. Make sure that any existing GlideRecord calls include a callback. For example, the following script does not include a callback and is incompatible with the mobile platform:

```javascript
var gr = new GlideRecord('incident');
gr.addQuery('number', g_form.getValue('related_incident'));
gr.query();
gr.next();
g_form.setValue('u_related_incident_description', gr.short_description);
```

The following script has been updated to include the callback, and is compatible with the mobile platform:

```javascript
var gr = new GlideRecord('incident');
gr.addQuery('number', g_form.getValue('related_incident'));
gr.query(function(gr) {
    gr.next();
    g_form.setValue('u_related_incident_description', gr.short_description);
});
```

| Do not use methods unavailable on the mobile platform | Due to the limitations and reduced functionality that is imposed by the mobile platform, the following methods are not deprecated but are not available on the mobile platform. If these run on the mobile platform, no action occurs:

- showRelatedList ()
- hideRelatedList ()
- showRelatedLists ()
- hideRelatedLists ()
- flash ()
- getSections ()
- enableAttachments ()
- disableAttachments ()
- setReadonly () (Note that setReadOnly () is available)
- getParameter ()

| Enable scripts for mobile | Scripts must be enabled for the mobile platform. See Enable client scripts for the mobile browser. |
UI policies

A field is now available on the UI Policy form for scripts to run on a desktop or mobile platform, or both at the same time. Update existing policies so that they apply to either the mobile platform or both. For new scripts, also ensure that the mobile option or both is selected. For more on UI policies for mobile, see Enable UI policies for the mobile browser.

Navigator modules

For existing code, modules must be transferred to either the sys_ui_application or sys_ui_module tables to be available on the mobile platform. When developing new code, be sure that all modules are created in the sys_ui_application or sys_ui_module tables. For more information, see Configure the mobile application navigator.

UI actions

UI actions must be transferred to the sys_ui_ng_action table to appear on the mobile platform. UI action scripts that do not use deprecated methods do not require changes to the script itself. For new UI actions, be sure that they are created in the sys_ui_ng_action table. For more information, see Mobile UI actions.

AJAX

AJAX (asynchronous JavaScript and XML) is a group of interrelated, client-side development techniques used to create asynchronous Web applications. AJAX enables web applications to send and retrieve information to and from a server in the background, without impacting the user experience with the displayed web page.

GlideAjax

The GlideAjax class allows the execution of server-side code from the client. GlideAjax calls pass parameters to the script includes, and, using naming conventions, allows the use of these parameters.

Note: This functionality requires a knowledge of JavaScript.

Using GlideAjax:

- Initialize GlideAjax with the name of the script include that you want to use.
- When creating the script include, you must set the name field to be exactly the same as the class name.
- When creating the script include, you must select the Client callable check box.
- Specify the parameter sysparm_name. GlideAjax uses sysparm_name to find which function to use.
- Any extra parameters may be passed in, all of which must begin with sysparm_. Avoid using predefined parameter names:
  - sysparm_name
  - sysparm_function
  - sysparm_value
  - sysparm_type
- Code is then executed with the getXML() or getXMLWait() functions.

For additional information on GlideAjax, refer to in the development portal.
Examples of asynchronous GlideAjax
There are two parts to the asynchronous GlideAjax script: client-side and server-side code.

Hello World: Returning a value from the server

Client side
This code runs on the client (the web browser). Create a client script as normal. This sends the parameters to server, which then does the processing. So that the client does not wait for the result, a callback function is used to return the result, passed to the getXML() function. (In this case it is called HelloWorldParse.)

The getXMLWait() function does not need a separate callback function, but this will block the client. If the client-server communication takes a long time (for example on slow networks), the application will seem unresponsive and slow. An example of getXMLWait() is in the following section.

```javascript
var ga = new GlideAjax('HelloWorld');
ga.addParam('sysparm_name', 'helloWorld');
ga.addParam('sysparm_user_name', 'Bob');
ga.getXML(HelloWorldParse);

function HelloWorldParse(response) {
    var answer = response.responseXML.documentElement.getAttribute("answer");
    alert(answer); }
```

Server side
The server-side code for the above function. Do not create a business rule, but instead navigate to System Definition > Script Include and create a new script. Paste in the code below.

Note: You must set the name of the script include to HelloWorld.

- The sys_script_include code must extend the AbstractAjaxProcessor class and be client-callable.
- Function names starting with '_' are considered private and are not callable from the client.
- Avoid overriding methods of AbstractAjaxProcessor, including initialize. While it is possible to invoke methods of your superclass object which you have overridden, it is complicated and best avoided altogether.

```javascript
var HelloWorld = Class.create();
HelloWorld.prototype =
Object.extendsObject(AbstractAjaxProcessor, {
    helloWorld:function() { return "Hello " +
        this.getParameter('sysparm_user_name') + "!";
    },
    _privateFunction: function() { // this function is not client callable
    }
});
```

This results in an alert box that says 'Hello Bob!' when you visit the form.
Returning multiple values

Since the response is an XML document we are not limited to returning a single answer value. Here is a more complex example returning multiple XML nodes and attributes.

AJAX processor script include

```javascript
/*
  * MyFavoritesAjax script include Description - sample AJAX processor returning multiple value pairs
  */
var MyFavoritesAjax = Class.create();
MyFavoritesAjax.prototype =
  Object.extendsObject(AbstractAjaxProcessor, {
    /*
    * method available to client scripts call using:
    * var ga = new GlideAjax("MyFavoritesAjax");
    * ga.addParam("sysparm_name","getFavorites");
    */
    getFavorites: function() { // build new response xml element for result
      var result = this.newItem("result");
      result.setAttribute("message","returning all favorites");
      //add some favorite nodes with name and value attributes
      this._addFavorite("color","blue");
      this._addFavorite("beer","lager");
      this._addFavorite("pet","dog");
    },
    // all items are returned to the client through the inherited methods of AbstractAjaxProcessor
    _addFavorite: function(name, value) {
      var favs = this.newItem("favorite");
      favs.setAttribute("name",name);
      favs.setAttribute("value",value);
    },
    type:"MyFavoritesAjax"
  });
```

Client script

```javascript
// new GlideAjax object referencing name of AJAX script include
var ga = new GlideAjax("MyFavoritesAjax");
// add name parameter to define which function we want to call
// method name in script include will be getFavorites
ga.addParam("sysparm_name","getFavorites");

// submit request to server, call ajaxResponse function with server response
ga.getXML(ajaxResponse);

function ajaxResponse(serverResponse) {
  // get result element and attributes
  var result = serverResponse.responseXML.getElementsByTagName("result")[0].getAttribute("message");
```
// check for message attribute and alert user
if(message) alert(message);

// build output to display on client for testing
var output = "";

// get favorite elements
var favorites = serverResponse.responseXML.getElementsByTagName("favorite");
for(var i = 0; i < favorites.length; i++) {
    var name = favorites[i].getAttribute("name");
    var value = favorites[i].getAttribute("value");
    output += name + " = " + value + "\n";
}

alert(output);

XML response

<xml sysparm_max="15" sysparm_name="getFavorites"
     sysparm_processor="MyFavoritesAjax">
   <result message = "returning all favorites"></result>
   <favorite name = "color" value = "blue"></favorite>
   <favorite name = "beer" value = "lager"></favorite>
   <favorite name = "pet" value = "dog"></favorite>
</xml>

Examples of synchronous GlideAjax
Use synchronous when your script cannot continue without the GlideAjax response. This stops the session until the response is received.

If your use case demands that no further processing can occur until the GlideAjax response has been received, you can use getXMLWait(). However, because this will slow down your code and lock the user session until the response is received, it is generally recommended that you use getXML() with a callback function.

Note: Do not use AJAXEvaluateSynchronously.

Note: The getXMLWait() method is not available in scoped applications.

This code results in a client-side alert that displays The Server Says Hello Bob!

The client code.

```javascript
var ga = new GlideAjax('HelloWorld')
ga.addParam('sysparm_name','helloWorld');
ga.addParam('sysparm_user_name','Bob');
ga.getXMLWait();
alert(ga.getAnswer());
```

The server-side script include code.

```javascript
var HelloWorld = Class.create();
HelloWorld.prototype = Object.extendObject(AbstractAjaxProcessor, { helloWorld: function() { return "The Server Says Hello " + this.getParameter('sysparm_user_name') + "!"; } });
```
AJAXClientHelper
Provides helper functions for Ajax clients to generate Choice Lists and retrieve displayed values from a choice list.

Where to use
Use this script include wherever your need to generate choice lists or retrieve a value from an Ajax client.

Method summary

<table>
<thead>
<tr>
<th>Method summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getValues()</td>
<td>Gets the values from the choice list.</td>
</tr>
<tr>
<td>generateChoice()</td>
<td>Generates the choices for a choice list.</td>
</tr>
<tr>
<td>generateChoiceTable()</td>
<td>Generates the choice table.</td>
</tr>
<tr>
<td>getDisplay()</td>
<td>Gets the display value from the choice list.</td>
</tr>
</tbody>
</table>

Method detail

<table>
<thead>
<tr>
<th>API method</th>
<th>Description</th>
<th>Input parameters</th>
<th>Output returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>getValues()</td>
<td>Gets the values from the choice list.</td>
<td>none</td>
<td>The choice list values.</td>
</tr>
<tr>
<td>generateChoice()</td>
<td>Generates the choices for a choice list.</td>
<td>none</td>
<td>The choice list.</td>
</tr>
<tr>
<td>generateChoiceTable()</td>
<td>Generates the choice table.</td>
<td>none</td>
<td>The choice table.</td>
</tr>
<tr>
<td>getDisplay()</td>
<td>Gets the display value from the choice list.</td>
<td>none</td>
<td>The display value.</td>
</tr>
</tbody>
</table>

AJAXClientTiming
Saves client timing values for a transaction.

Method summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>process()</td>
<td>Method called by the Prototype JavaScript Framework during object processing.</td>
</tr>
</tbody>
</table>
Method detail

<table>
<thead>
<tr>
<th>Method Detail</th>
<th>Description</th>
<th>Input Fields</th>
<th>Output Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>process()</td>
<td>Called by the Prototype JavaScript Framework during object processing. Do not call this method directly.</td>
<td>none</td>
<td>Returns: void.</td>
</tr>
</tbody>
</table>

Jelly tags

This is a list of Jelly tags and Glide tags.

Jelly Tags

if

- **Description**: The *if* tag is just what it looks like, an if tag. This is like an if statement in any programming language, but keep in mind that there is no elseif tag and no else tag. If you want to create that kind of structure, try the choose/when/otherwise syntax.
- **Parameters**: **test** - The condition to evaluate in order to determine if the block will execute.
- **Example**:

  ```
  <g:evaluate var="jvar_gr" object="true">
  var gr = new GlideRecord("incident");
  gr.addQuery("active", true);
  gr.query();
  gr;
  </g:evaluate>

  <j:if test="!jvar_gr.hasNext()">
  We did not find any active incidents.
  </j:if>
  <j:if test="jvar_gr.next()">
  We found ${jvar_gr.getRowCount()} active incidents.
  </j:if>
  ```

while

- **Description**: The *while* tag does a while loop.
- **Parameters**: **test** - The condition to evaluate in order to determine if the statement will loop through. This should be an expression enclosed in ${} or $() that evaluates to true or false.
- **Example**:

  ```
  <g:evaluate var="jvar_gr" object="true">
  var gr = new GlideRecord("incident");
  gr.addQuery("active", true);
  gr.query();
  gr;
  </g:evaluate>

  <j:while test="jvar_gr.next()">
  <a href="incident.do?sys_id=${jvar_gr.getValue('sys_id')}">
  ${jvar_gr.getValue('number')}<a/>
  </j:while>
  ```
set

- **Description:** The set tag sets a variable.
- **Parameters:**
  - **var** - The variable to set. Often the system prefixes these variables with jvar_ for consistency.
  - **value** - The value to set var to. This is often an expression enclosed in ${} or $().
  - **defaultValue** - If the value results to null or empty, this value is put into the var.
- **Example:**

```xml
<j:set var="jvar_incident_number" value="${jvar_gr.getValue('number')}"/>
```

set_if

- **Description:** The set_if tag sets a variable based on a test. This tag is similar to the ternary operator in other programming languages (var = <test> ? <if_true> : <if_false>).
- **Parameters:**
  - **var** - The variable to set. Often the system prefixes these variables with jvar_ for consistency.
  - **test** - The condition to evaluate in order to determine if the statement will evaluate the true value or the false value. This should be an expression enclosed in ${} or $() that evaluates to true or false.
  - **true** - The value to set the variable to if test evaluates to true. This parameter is optional, so if the field is blank, and if test evaluates to true, the variable will be left blank.
  - **false** - The value to set the variable to if test evaluates to false. This parameter is optional, so if the field is blank, and if test evaluates to false, the variable will be left blank.

choose

- **Description:** The choose tag starts a choose block of code. This is similar to the if-elseif-else kind of syntax in most programming languages. With a choose tag, you can use when and otherwise tags to specify other blocks of code.
- **Parameters:** None.
- **Example:**

```xml
<j:choose>
  <j:when test="${jvar_gr.getRowCount() lt; 1}">We found multiple records!</j:when>
  <j:when test="${jvar_gr.next()}">We found record ${jvar_gr.getValue('number')}\</j:when>
  <j:otherwise>Sorry, we could not find the record you specified.</j:otherwise>
</j:choose>
```

when

- **Description:** The when tag is used within a choose block to indicate a condition. This tag is similar to an if or an elseif in that it specifies a condition, executes the inner content, and then implies a break at the end to leave the if-elseif construct.
- **Parameters**: 
  - **test**: The condition to evaluate in order to determine if the statement will loop through. This should be an expression enclosed in ${()}$ or $[()$] that evaluates to true or false.
  - **Example**:

    ```xml
    <j:choose>
      <j:when test="${jvar_gr.getRowCount()} {AMP}lt; 1">We found multiple records!</j:when>
      <j:when test="${jvar_gr.next()}">We found record ${jvar_gr.getValue('number')}\</j:when>
      <j:otherwise>Sorry, we could not find the record you specified.</j:otherwise>
    </j:choose>
    ``

- **otherwise**

  - **Description**: The `otherwise` tag is used within a `choose/when/otherwise` block, and is like the "else" or "default" case.
  - **Parameters**: None.
  - **Example**:

    ```xml
    <j:choose>
      <j:when test="${jvar_gr.getRowCount()} {AMP}lt; 1">We found multiple records!</j:when>
      <j:when test="${jvar_gr.next()}">We found record ${jvar_gr.getValue('number')}\</j:when>
      <j:otherwise>Sorry, we could not find the record you specified.</j:otherwise>
    </j:choose>
    ```

**Glide Tags**

- **evaluate**

  - **Description**: The `evaluate` tag evaluates JavaScript code (server side), and makes variables visible to future code. Unlike other tags, the `evaluate` tag evaluates the content that is inside the tag as server side JavaScript.

    The context is the same as that of script includes in the system. Other script includes, global business rules, GlideRecord, GlideSystem, and Jelly variables (prefixed with `jelly.` if the parameter `jelly="true"` is set) are available.

  - **Parameters**:
    - **var**: The name of the variable that will be set to the result of the script.
    - **object**: If set to `true`, the result of the expression will be treated as an object instead of a primitive variable (string or integer variable values).
    - **jelly**: If set to `true`, allows Jelly context variables to be referenced in the script.
    - **expression**: This is an expression to be executed for the value to put in `var`. The expression can be either of two places. First, it can be an attribute on the `evaluate` tag itself. Otherwise, the content between the beginning tag and ending tag is the expression. The last line of the expression is the actual value passed into `var`.

  - **Example**:

    ```xml
    <g:evaluate var="jvar_gr" object="true">
      var gr = new GlideRecord("incident");
    </g:evaluate>
    ```
messages

- **Description:** The `messages` tag helps with translation. When `gs.getMessage()` is called anywhere on a page, there are two possible places where the translation is found. First, the page checks a local cache of translations. Second, the page makes an AJAX call to the server to find the translation. What `g:messages` does is allow pages to cache certain messages.
- **Parameters:** None.
- **Example:**

```xml
<g:messages>
  Yes
  No
  Cancel
</g:messages>
```

breakpoint

- **Description:** When the `breakpoint` tag is called, it prints a list of all the variables in Jelly at the current moment, with their respective values. If a variable is specified, it prints the requested variable and its value. The output is placed in the System Log.
- **Parameters:** `var` - (Optional) The variable to log the value for. If `var` is not specified, then all variables will be dumped into the log.
- **Example:**

```xml
<g:breakpoint />

<g:breakpoint var="sysparm_view"/>
```

no_escape

- **Description:** The system, by default, uses escaped output as a security measure. Output placed inside of `no_escape` tags is not escaped before output. Be careful when using these tags, because if user input is displayed here it can open a security vulnerability on the page.
- **Parameters:** None.
- **Example:**

```xml
<g:no_escape>${jvar_raw_html_data}</g:no_escape>
```

macro_invoke
• **Description:** The `macro_invoke` tag calls a UI macro that you have specified in the database. You may also call a UI macro by specifying it in the tag name. For example, if you had a UI macro named `my_macro`, you could call that macro with the tag `<g:my_macro/>`.

• **Parameters:**
  
  • `macro` - The name of the UI macro to execute. If your tag name is `g:macro_invoke`, then the macro attribute specifies the name of the macro. If the tag name includes the name of the macro, then there is no need to include a macro attribute.
  
  • Other attributes - For each attribute you specify, a variable with that name will be available in the context of the UI macro, prefixed with `jvar_`.

• **Example:**

```xml
<!-- Will invoke the contents of the UI macro named "sample_macro", which will have the variable jvar_message available within it-->
<g:macro_invoke macro="sample_macro" message="This is a sample macro variable." />
```

```xml
<!-- Will invoke the contents of the UI macro named "sample_macro", which will have the variable jvar_message available within it-->
<g:sample_macro message="This is a sample macro variable." />
```

---

**Jelly escaping types**

You use different methods when escaping characters in JavaScript and HTML. JavaScript uses the backslash character, and HTML uses the ampersand character.

**Note:** This functionality requires a knowledge of JavaScript, HTML, and Apache Jelly (a Java and XML based scripting and processing engine for turning XML into executable code).

There are two different types of escaping that is required when generating output from Jelly:

• JavaScript

• HTML

The escaping for each of these consists of:

<table>
<thead>
<tr>
<th>Type</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript</td>
<td>' (single quote)</td>
<td>\</td>
</tr>
<tr>
<td></td>
<td>&quot; (double quote)</td>
<td>\</td>
</tr>
<tr>
<td></td>
<td>CR (carriage return)</td>
<td>(blank)</td>
</tr>
<tr>
<td></td>
<td>NL (newline)</td>
<td>\n (' followed by 'n')</td>
</tr>
<tr>
<td>HTML</td>
<td>&amp; (ampersand)</td>
<td>&amp;</td>
</tr>
<tr>
<td></td>
<td>&lt; (less than)</td>
<td>&lt;</td>
</tr>
<tr>
<td></td>
<td>&gt; (greater than)</td>
<td>&gt;</td>
</tr>
</tbody>
</table>
You can also escape HTML using the getHTMLValue() function which will enforce all line breaks and escape the characters mentioned above. It can be used as follows:

```
${test.getHTMLValue()}
```

Add escaping to a Jelly replacement
You can handle character escaping in Jelly files. XML escaping behavior can be modified only by users with the security_admin role.

**Note:** This functionality requires a knowledge of JavaScript, HTML, and Apache Jelly (a Java and XML based scripting and processing engine for turning XML into executable code).

Add a prefix to the `expression` or `[expression]` indicating the escaping to be performed.

```
${JS:expression}
${HTML:expression}
```

The prefix tells the system to take the result of the expression and escape it before outputting. The escaping may be combined by specifying a comma-separated list of prefixes:

```
${JS,HTML:expression}
```

**Extensions to Jelly syntax**
Apache's Jelly syntax is used to render forms, lists, UI pages, and many other things rendered in ServiceNow.

With Jelly, logic can be embedded within static content and computed values may be inserted into the static content.

**Attention:** This functionality requires a knowledge of Apache Jelly (a Java and XML based scripting and processing engine for turning XML into executable code).

This page from Apache has a summary of the standard Jelly tags: [http://commons.apache.org/jelly/tags.html](http://commons.apache.org/jelly/tags.html)

**Namespaces**
Jelly often includes multiple namespaces when invoking tags.

The "j" namespaces are standard Jelly whereas the "g" namespaces are unique to ServiceNow scripts. For example, the `<g:evaluate>` tag is supplied by ServiceNow to allow you to compute a value using JavaScript. The standard Jelly tag `<j:test>` is used to evaluate a condition.

**Phases**
Usually, there are two phases indicated by namespaces `<j>` versus `<j2>` and `<g>` versus `<g2>`.

The namespaces without the "2" happen in the first phase of processing and these are cached except when used in a UI page. Those with the '2' are never cached. Care must be taken when selecting whether to use phase 1 or phase 2 for efficiency and correct results.

In addition to the namespaces, the syntax used to insert values into static content differs depending on which phase is to supply the value. A dollar with braces surrounding a value inserts the value in phase 1. For example, `${jvar_ref}` inserts the value `jvar_ref` during phase 1 of the jelly process. A dollar with brackets surrounding a value inserts the value in phase 2. For example,
${jvar_ref} inserts the value jvar_ref during phase 2. A value surrounded by quotes is treated as a string. For example, '${jvar_ref}' inserts the value jvar_ref as a string during phase 2.

```<script>
if (confirm("${gs.getMessage('home.delete.confirm')}"))
...
</script>
```

```<input type="hidden" id="${jvar_name}" name="${jvar_name}"
value="${jvar_value}" class="${jvar_class}" />
```

If tests
You can use if statements in Jelly scripts.

Testing whether something is true or not can be done as follows:

```<j:if test="${jvar_something}">...do something...</j:if>
<j:if test="${!jvar_something}">...do something...</j:if>
```

The reason this statement works, is that, in Jelly, a term like jvar_something is "truthful" in an if tag if:
1. it is Boolean and true
2. it is a String and = "true", "yes", "on", or "1"

Testing whether something exists can be done as follows:

```<j:if test="${empty(jvar_something)}">...do something...</j:if>
```

The reason this statement works is that the JEXL empty function returns true if its argument is:
1. null
2. an empty string
3. a zero length collection
4. a map with no keys
5. an empty array

**Note:** You cannot mix javascript and jvar variables in a JEXL expression. They must be broken into separate expressions.

Set If
Sets a variable to one of two different values depending on whether a test is true or false.

```<g2:set_if var="jvar_style" test="${gs.getPreference('table.compact')} != 'false'">
true="margin-top:0px; margin-bottom:0px;"
false="margin-top:2px; margin-bottom:2px;" /
```

<g:insert> versus <g:inline> versus <g:call>
This page provides a comparative explanation of three tags: <g:insert>, <g:inline>, and <g:call>.
<g:insert>
The <g:insert> tag inserts a Jelly file into your Jelly in a new context. This means you cannot access the variables previously established in your Jelly.

```xml
<g:insert template="get_target_form_function.xml"/>
```

<g:inline>
The <g:inline> tag inserts a Jelly file into your Jelly in the same context. This means that the inserted Jelly can access the variables you previously established and it can change the values of those variables.

```xml
<g:inline template="element_default.xml"/>
```

<g:call>
For better encapsulation, the <g:call> tag may be used. Your function will only have access to the values passed to it. The Jelly context will look the same after a call as before the call. This means you cannot set a global variable here and read it later. This also means you can’t mistakenly set a global variable called “jvar_temp” and overwrite a variable that somebody else was relying on.

Passing values, if needed, is done explicitly by including the name of the parameter on the <g:call> line followed by the equal sign followed by the value in quotes:

```xml
<g:call function="collapsing_image.xml" id="${jvar_section_id}"
   image="${jvar_cimg}"
   first_section_id="${jvar_first_section_id}" image_alt="${jvar_cimg_alt}"/>
```

If values are passed, and you want to have defaults or required parameters, your Jelly referenced in the function must then include a line to declare whether the parameters are required or have a default value:

```xml
<g:function id="REQUIRED" image="REQUIRED" image_prefix=""
   image_alt="REQUIRED"/>
```

The example above indicates that 3 of the parameter are required and one parameter is option with a blank default value. Note that if you are not passing values or if you do want to have default or required values, you do not need to include the <g:function> line at all. In general, however, you will want to include a <g:function> line.

The value can then be referenced in your template by prepending the “jvar_” prefix to the parameter’s name:

```xml
<img id="img.$\{jvar_id\}" src="images/$\{jvar_image\}" alt="$\{jvar_image_alt\}"
   onclick="toggleSectionDisplay('$\{jvar_id\}',$\{jvar_image_prefix\},'$\{jvar_first_section_id\}');"/>
```

For <g:call>, parameters may also be pass implicitly as a list of named variables in an “arguments” parameter:

```xml
<g:call function="item_link_default.xml"
   arguments="sysparm_view,ref_parent,jvar_target_text"/>
```
As an alternative to passing variables into the function via separate tag arguments, it is possible to pass a list of variables in a single 'arguments' argument. All variables identified by name (comma separated) in the argument parameter are re-introduced within the function under the exact same name (e.g. inside the function template, we’d have variables sysparm_view, ref_parent, and jvar_target_text available to us).

The function template may return a value to the calling template using the return= attribute. Within the function the jvar_answer variable sets the return value.

```
<g:call function="item_body_cell_calc_style.xml" arguments="jvar_type" return="jvar_style"/>
```

The <g:evaluate> tag is used to evaluate an expression written in Rhino JavaScript and sometimes to set a variable to the value of the expression.

The last statement in the expression is the value the variable will contain.

```
<g:evaluate var="jvar_page" jelly="true">
    var page = "";
    var pageTitle = "";
    var pageGR = new GlideRecord("cmn_schedule_page");
    pageGR.addQuery("type", jelly.jvar_type);
    pageGR.query();
    if (pageGR.next()) {
        page = pageGR.getValue("sys_id");
        pageTitle = pageGR.getDisplayValue();
    }
    page;
</g:evaluate>
```

```
<g:evaluate var="not_important" expression="sc_req_item.popCurrent()" />
```

If you would like to have the evaluate return an object (for example an array), use the argument object="true".

```
<g:evaluate object="true" var="jvar_items" expression="SncRelationships.getCMDBViews()" />
```

If you would like to access Jelly variables inside an evaluate, include jelly="true" in the evaluate and add "jelly." before the Jelly variable's name. For example, to access the GlideJellyContext:

```
<g:evaluate var="jvar_row_no" jelly="true">
    var gf = jelly.context.getGlideForm();
    var row = gf.getRowNumber();
    row;
</g:evaluate>
```

Another example of accessing a jvar using the jelly="true" parameter. The value of jvar_h was set previously and we can access it inside the evaluate:

```
[$NLBR:jvar_h.getHTMLValue('newvalue')]
```
copyToPhase2="true"

If you have a need to take the results of an evaluation that occurs in phase 1 and propagate it to phase 2, use copyToPhase2="true". There is some protection for escaping in this use. For example:

```xml
<g:evaluate var="jvar_has_special_inc" copyToPhase2="true">
  var specialInc = gs.tableExists("special_incident");
  specialInc;
</g:evaluate>

[$[jvar_has_special_inc]
```

If you do not need to evaluate something, you can do this more directly. Beware of escaping issues here (double quotes in jvar_rows would cause a problem in the example):

```xml
<j2:set var="jvar_rows" value="${jvar_rows}"/>
```

This tag can be used to display the current Jelly variables and their values in the log.

Be sure to remove this tag before going to production.

This tag defines a form on the UI page.

For example, if your form contained the application_sys_id field:

```xml
<g:ui_form>
  <p>Click OK to run the processing script.</p>
  <g:dialog_buttons_ok_cancel ok="return true" />
  <input type="hidden" name="application_sys_id" value="499836460a0a0b1700003e7ad950b5da" />
</g:ui_form>
```

The g:ui_form may benefit greatly from a processing script.

This tag adds a reference to a UI macro that creates an input field on a page that allows users to input information. The ui_input_field passes a label, name, value, and size into the UI macro.

Here is an example from a UI page:

```xml
<g:ui_input_field label="sys_id" name="sysid" value="9d385017c611228701d22104cc95c371" size="50"/>
```

This tag puts a user-editable check mark on a page. The name and value are passed into the UI macro.

Here is an example from a table on a UI page:

```xml
<table>
  <tr>
    <td nowrap="true">
      <label>Time Card Active:</label>
    </td>
```
This tag puts buttons on the UI page that run a specified processing script if the tag returns true.

If your UI page contains a form (uses the <g:form> tag), you can submit the form and have the Processing Script run. The Processing Script can naturally access fields on the form. For example, if your form contained the application_sys_id field:

```html
<g:ui_form>
  <p>Click OK to run the processing script.</p>
  <g:dialog_buttons_ok_cancel ok="return true" />
  <input type="hidden" name="application_sys_id" value="499836460a0a0b1700003e7ad950b5da"/>
</g:ui_form>
```

This tag adds a reference to a page that can be referenced by a Processing Script.

The following example creates a reference defined by name, id, and table parameters in the tag:

```html
<g:ui_reference name="QUERY:active=true^roles=itil" id="assigned_to" table="sys_user" />
```

Then in the Processing Script, reference the name field like this:

```javascript
newTask.assigned_to = request.getParameter("QUERY:active=true^roles=itil");
```

You can specify a reference qualifier, so that the ‘name’ attribute can be unique. The following example creates a reference defined by name, id, and table parameters in the tag. Note: the ‘columns’ attribute only applies to the auto-completer.

```html
<g:ui_reference name="parent_id" id="parent_id" table="pm_project" query="active=true" completer="AJAXTableCompleter" columns="project_manager;short_description"/>
```

Ampersand

Ampersands in Jelly can cause you grief because Jelly is XML.

Use ${AMP} to insert an ampersand in Jelly. If you are writing JavaScript that appears in the HTML part of say a UI page or UI macro that is actually going to run on the browser you are better off putting this code in the “client script” field and that way you can avoid escaping issues. However, if you really must put it in the “HTML” field, you will need to do something like this:

```javascript
ta = ta[1].split('${AMP}');
```

And

Use ${AND} to insert a JavaScript and in Jelly.

For example:

```javascript
if (d ${AND} e) {
  var color = d.value;
}
Alternately, in a Jelly test you would use `&amp;&amp;`. For example:

```xml
<j:jif test="${jvar_form_name} == 'sys_form_template' &amp;&amp; ! RP.isDialog()">"
```

Less than

Similar to ampersands, less than (`"<"`) signs can also cause problems due to Jelly being XML. This can be resolved by reversing your test such that it is not necessary or by using `&lt;` in place of the less than sign.

```xml
<g2:evaluate var="jvar_text">
  var days = "";
  var selectedDays = '${${ref}}';
  for (var i = 1; i &lt;= 7; i++) {
    if (selectedDays.indexOf(i.toString()) &gt;= 0) {
      days += gs.getMessage("dow" + i);
      days += " ";
    }
  }
  days;
</g2:evaluate>
```

Many times you can avoid the “less than” operator all together by just using 'not equals' which doesn’t have escaping issues. For example:

```xml
for (var i=0; i != ta.length; i++) {
}
```

Whitespace

Normally, white space is removed by Jelly. To keep it, you must specify that it not be trimmed.

For example, the following keeps the space after the colon.

```xml
<j2:whitespace trim="false">${gs.getMessage('Did you mean')}: </j2:whitespace>
```

Spaces

To encode a non-breaking space (&nbsp;), you can use `&SP;`.

For example:

```xml
<span id="gsft_domain" style="display: inline">${gs.getMessage('Domain')};&SP;
  <span id="domainDD" class="drop_down_element" style="text-decoration: none; color: white">${gs.getMessage("Loading..."))
</span>
</span>
```

Tracing Jelly

ServiceNow has a feature that allows the evaluation of Jelly to be traced.

The trace is sent to the log. This should only be turned on during debugging as this produces a lot of logging. To turn on the trace, set the property glide.ui.template.trace to true. For example, the following script can be executed to do this:

```xml
GlideProperties.set ( 'glide.ui.template.trace' , true ) ;
```

If you want to see your log entries on your web browser at the bottom of each page, navigate to `System Diagnostics > Debug Log`.
Server API reference

Server-side Glide APIs enable you to create scripts to run on the server. Script includes provide common functionality that you can include in your server-side scripts. This page provides links to reference information for the server-side APIs and script includes.

### Glide API reference

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMDBUtil</td>
<td>Provides utility methods for creating and managing table relationships in the configuration management database (CMDB) and managing CMDB baselines.</td>
</tr>
<tr>
<td>ExpenseLine</td>
<td>Provides methods for generating expense line (fm_expense_line) records from your own server-side scripts for Cost Management.</td>
</tr>
<tr>
<td>GlideAggregate</td>
<td>Extends GlideRecord to allow database aggregation queries, such COUNT, SUM, MIN, MAX, and AVG, for creating customized reports or calculations in calculated fields.</td>
</tr>
<tr>
<td>GlideDateTime</td>
<td>Performs date-time operations, such as date-time calculations, formatting a date-time, or converting between date-time formats.</td>
</tr>
<tr>
<td>GlideElement</td>
<td>Provides methods to operate on the fields of the current GlideRecord.</td>
</tr>
<tr>
<td>GlideRecord</td>
<td>Provides methods for performing database operations. GlideRecord is a special Java class that can be used in JavaScript as if it were a native JavaScript class, instead of writing SQL queries.</td>
</tr>
<tr>
<td>GlideRecordSecure</td>
<td>Provides GlideRecord methods that enforce access control lists (ACLs).</td>
</tr>
<tr>
<td>GlideSystem</td>
<td>Provides methods to obtain information about the system.</td>
</tr>
<tr>
<td>GlideTimeline</td>
<td>Provides methods for configuring and displaying a Glide Windowing Toolkit Timeline when customizing timeline schedule pages.</td>
</tr>
<tr>
<td>Service Catalog</td>
<td>Provides methods for accessing service catalog configuration and actions from within the Service Portal.</td>
</tr>
</tbody>
</table>
**Script Includes reference**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Collision</td>
<td>Provides helper functions useful for scripting Change Collision on the server-side or when using AJAX calls on the client.</td>
</tr>
<tr>
<td>Discovery</td>
<td>Provides functions to accomplish common Discovery tasks.</td>
</tr>
<tr>
<td>JavaScript Tools</td>
<td>Provides general JavaScript utility functions that are useful for tasks such as logging, validating scripts, and exception wrapping.</td>
</tr>
<tr>
<td>MID Server</td>
<td>Provides functions to accomplish common MID Server tasks.</td>
</tr>
<tr>
<td>Utility</td>
<td>Provides general utility functions that are useful when working with arrays, datetimes, and tables.</td>
</tr>
<tr>
<td>Workflow</td>
<td>Provides functions that are useful when working with workflows.</td>
</tr>
</tbody>
</table>

**Notify API reference**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notify API</td>
<td>Provides functionality for sending SMS messages and setting up conference calls.</td>
</tr>
</tbody>
</table>

**Glide class overview**

The ServiceNow Glide classes expose JavaScript APIs that enable you to conveniently work with tables using scripts.

Using the Glide APIs, you can perform database operations without writing SQL queries, display UI pages, and define UI actions. The following tables provide brief descriptions of the Glide classes and links to detailed information.

**Server-side Glide classes**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideRecord</td>
<td>Use this class for database operations instead of writing SQL queries. GlideRecord is a special Java class that can be used in JavaScript exactly as if it were a native JavaScript class. A GlideRecord is an object that contains records from a single table. See .</td>
</tr>
<tr>
<td>GlideElement</td>
<td>Use this class to operate on the fields of the current GlideRecord. See .</td>
</tr>
<tr>
<td>GlideSystem</td>
<td>Use this class to get information about the system. See .</td>
</tr>
</tbody>
</table>
### Client-side Glide Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideAggregate</td>
<td>Use this class to perform database aggregation queries, such as COUNT, SUM,</td>
</tr>
<tr>
<td></td>
<td>MIN, MAX, and AVG, for creating customized reports or calculations in</td>
</tr>
<tr>
<td></td>
<td>calculated fields. See .</td>
</tr>
<tr>
<td>GlideDateTime</td>
<td>Use this class to perform date-time operations, such as date-time calcula-</td>
</tr>
<tr>
<td></td>
<td>tions, formatting a date-time, or converting between date-time formats. See</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
</tbody>
</table>

**Glide stack**

Glide is an extensible Web 2.0 development platform written in Java that facilitates rapid development of forms-based workflow applications (work orders, trouble ticketing, and project management, for example).

**User interface stack technology map**

<table>
<thead>
<tr>
<th>Java packages</th>
<th>Technologies used</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>- AngularJS</td>
</tr>
<tr>
<td>(Browser)</td>
<td>- HTML</td>
</tr>
<tr>
<td></td>
<td>- CSS</td>
</tr>
<tr>
<td></td>
<td>- JavaScript</td>
</tr>
<tr>
<td>com.glide.ui</td>
<td>GlideServlet</td>
</tr>
<tr>
<td>com.glide.jelly</td>
<td>Apache Jelly</td>
</tr>
<tr>
<td>com.glide.script</td>
<td>Business Rules</td>
</tr>
<tr>
<td>com.glide.db</td>
<td>Persistence</td>
</tr>
<tr>
<td></td>
<td>Mozilla Rhino</td>
</tr>
<tr>
<td></td>
<td>JDBC</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
### User interface stack technology map descriptions

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Attributes</th>
</tr>
</thead>
</table>
| GlideServlet     | The primary driver of Glide, and the only servlet in the system, is found in GlideServlet.java. The GlideServlet:                                                                                                                                                                                                                       | • Handles inbound action requests  
• Renders pages  
• Merges data with forms  
• Presents to user  
• Interfaces with script layer |
| Business Rules   |                                                                                                                                                                                                                                                                                                                                           | • ECMA / JavaScript implementation based on Mozilla Rhino  
• Interfaces with persistence layer using JDBC recordset interface  
• Merges persistence layer meta-data with data for easy correlation |
| Persistence      |                                                                                                                                                                                                                                                                                                                                           | • Persistence means any store  
• RDBMS  
• LDAP  
• File system  
• Uniform access regardless of store type  
• Provides QUID and meta-data capabilities  
• Interfaces presented to callers  
• RecordSet  
• TableDescriptor  
• ElementDescriptor |

© 2020 ServiceNow, Inc. All rights reserved.  
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Glide servlet

Scripting alert, info, and error messages
You can send messages to customers as alerts, informative messages, or error messages.

Business rule and other general use scripts

<table>
<thead>
<tr>
<th>Script</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>current.field_name.setError(&quot;Hello World&quot;);</code></td>
<td>Will put &quot;Hello World&quot; below the specified field.</td>
</tr>
<tr>
<td><code>gs.addInfoMessage(&quot;Hello World&quot;);</code></td>
<td>Will put &quot;Hello World&quot; on the top of the screen.</td>
</tr>
<tr>
<td><code>gs.print(&quot;Hello World&quot;);</code></td>
<td>Will write to the text log on the file system but not to the sys_log table in the database.</td>
</tr>
</tbody>
</table>
### Script

<table>
<thead>
<tr>
<th>Script</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gs.log(&quot;Hello World&quot;);</code></td>
<td>Will write to the database and the log file.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Too much of this can adversely affect performance.</td>
</tr>
</tbody>
</table>

### Important:
The methods in this table are only for use in client scripts.

### Client side scripts

<table>
<thead>
<tr>
<th>Script</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>alert(&quot;Hello World&quot;);</code></td>
<td>Will pop up a window with “Hello World” and an ‘OK’ button.</td>
</tr>
<tr>
<td><code>confirm(&quot;Hello World&quot;);</code></td>
<td>Will pop up a window with “Hello World?” and a ‘Ok’ and ‘Cancel’ buttons.</td>
</tr>
<tr>
<td><code>g_form.showFieldMsg(&quot;field_name&quot;, &quot;Hello World&quot;, &quot;error&quot;);</code></td>
<td>Puts “Hello World” in an error message below the specified field.</td>
</tr>
<tr>
<td><code>g_form.hideFieldMsg(&quot;field_name&quot;);</code></td>
<td>Hides an error message that is visible under the specified field.</td>
</tr>
</tbody>
</table>

It is also possible to add other custom messages to your forms if necessary using client scripting. The text size of info and error messages at the top of the screen is customizable. Two properties control this. If you configured your forms, you may need to add these properties.

### Error and alert text size properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>css.outputmsg.info.text.font-size</code></td>
<td>Sets the size for info messages. Default is 11pt.</td>
</tr>
<tr>
<td><code>css.outputmsg.error.text.font-size</code></td>
<td>Sets the size for error messages. Default is 11pt.</td>
</tr>
</tbody>
</table>

### Glide Server APIs

ServiceNow provides APIs for the Glide Server.

**GlideAggregate**

The GlideAggregate class is an extension of GlideRecord and allows database aggregation (COUNT, SUM, MIN, MAX, AVG) queries to be done. This can be helpful in creating customized reports or in calculations for calculated fields.

**Note:** This functionality requires a knowledge of JavaScript.

For additional information, refer to API.

GlideAggregate examples

GlideAggregate is an extension of GlideRecord and its use is probably best shown through a series of examples.
Note: This functionality requires a knowledge of JavaScript.

Here is an example that simply gets a count of the number of records in a table:

```javascript
var count = new GlideAggregate('incident');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if(count.next())
  incidents = count.getAggregate('COUNT');
```

There is no query associated with the preceding example. If you want to get a count of the incidents that were open, simply add a query as is done with GlideRecord. Here is an example to get a count of the number of active incidents.

```javascript
var count = new GlideAggregate('incident');
count.addQuery('active','true');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if(count.next())
  incidents = count.getAggregate('COUNT');
```

To get a count of all the open incidents by category the code is:

```javascript
var count = new GlideAggregate('incident');
count.addQuery('active','true');
count.addAggregate('COUNT','category');
count.query();
while(count.next()){
  var category = count.category;
  var categoryCount = count.getAggregate('COUNT','category');
  gs.log("The are currently "+categoryCount+" incidents with a category of "+category);
}
```

The output is:

```plaintext
*** Script: The are currently 1.0 incidents with a category of Data
*** Script: The are currently 11.0 incidents with a category of Enhancement
*** Script: The are currently 1.0 incidents with a category of Implementation
*** Script: The are currently 197.0 incidents with a category of inquiry
*** Script: The are currently 13.0 incidents with a category of Issue
*** Script: The are currently 1.0 incidents with a category of request

The following is an example that uses multiple aggregations to see how many times records have been modified using the MIN, MAX, and AVG values.

```javascript
var count = new GlideAggregate('incident');
count.addAggregate('MIN','sys_mod_count');
count.addAggregate('MAX','sys_mod_count');
count.addAggregate('AVG','sys_mod_count');
count.groupBy('category');
count.query();
while(count.next()){
  var min = count.getAggregate('MIN','sys_mod_count');
  var max = count.getAggregate('MAX','sys_mod_count');
  var avg = count.getAggregate('AVG','sys_mod_count');
  gs.log("The number of times modified were Min: "+min+" Max: "+max+" Avg: "+avg);
}
```
var max = count.getAggregate('MAX','sys_mod_count');
var avg = count.getAggregate('AVG','sys_mod_count');
var category = count.category.getDisplayValue();
gs.log(category +" Update counts: MIN = "+ min +" MAX = "+ max +" AVG = "+ avg);}

The output is:

*** Script: Data Import Update counts: MIN = 4.0 MAX = 21.0 AVG = 9.3333
*** Script: Enhancement Update counts: MIN = 1.0 MAX = 44.0 AVG = 9.6711
*** Script: Implementation Update counts: MIN = 4.0 MAX = 8.0 AVG = 6.0
*** Script: inquiry Update counts: MIN = 0.0 MAX = 60.0 AVG = 5.9715
*** Script: Inquiry / Help Update counts: MIN = 1.0 MAX = 3.0 AVG = 2.0
*** Script: Issue Update counts: MIN = 0.0 MAX = 63.0 AVG = 14.9459
*** Script: Monitor Update counts: MIN = 0.0 MAX = 63.0 AVG = 3.6561
*** Script: request Update counts: MIN = 0.0 MAX = 53.0 AVG = 5.0987

The following is a more complex example that shows how to compare activity from one month to the next.

var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count','category');
agg.orderBy('category');
agg.addQuery('opened_at','>=','javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at','<=','javascript:gs.monthsAgoEnd(2)');
agg.query();
while(agg.next()){
  var category = agg.category;
  var count = agg.getAggregate('count','category');
  var query = agg.getQuery();
  var agg2 = new GlideAggregate('incident');
  agg2.addAggregate('count','category');
  agg2.orderByAggregate('count','category');
  agg2.orderBy('category');
  agg2.addQuery('opened_at','>=','javascript:gs.monthsAgoStart(3)');
  agg2.addQuery('opened_at','<=','javascript:gs.monthsAgoEnd(3)');
  agg2.addEncodedQuery(query);
  agg2.query();
  var last ="";
  while(agg2.next()){
    last = agg2.getAggregate('count','category');
  }
  gs.log(category +": Last month:"+ count +" Previous Month:"+ last);
}

The output is:

*** Script: Monitor: Last month:6866.0 Previous Month:4468.0
*** Script: inquiry: Last month:142.0 Previous Month:177.0
*** Script: request: Last month:105.0 Previous Month:26.0
*** Script: Issue: Last month:8.0 Previous Month:7.0
*** Script: Enhancement: Last month:5.0 Previous Month:5.0
*** Script: Implementation: Last month:1.0 Previous Month:0

The following is an example to obtain distinct count of a field on a group query.

var agg = new GlideAggregate('incident');
agg.addAggregate('count');
agg.addAggregate('count(distinct','category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
//
agg.groupBy('priority');
agg.query();
while (agg.next()) {
  // Expected count of incidents and count of categories within each priority
  // value (group)
  gs.info('Incidents in priority ' + agg.priority + ' = ' +
  agg.getAggregate('count') +
  ' (' + agg.getAggregate('count(distinct','category') + ' categories)');
}

The output is:

*** Script: Incidents in priority 1 = 13 (3 categories)
*** Script: Incidents in priority 2 = 10 (5 categories)
*** Script: Incidents in priority 3 = 5 (3 categories)
*** Script: Incidents in priority 4 = 22 (6 categories)

You can implement the SUM aggregate with or without the use of the groupBy() method. If you do not use the groupBy() method, the result of the SUM is the cumulative value for each different value of the field for which you request the SUM. For example, if you SUM the total_cost field in the Fixed Asset table, and the Fixed Asset table contains 12 total records:

- Three records with a total_cost of $12
- Four records with a total_cost of $10
- Five records with a total_cost of $5

When you SUM the record set, the getAggregate() method returns three different sums: $36, $40, and $25.

The following code illustrates implementing the SUM aggregate without using the groupBy() method:

```javascript
var totalCostSum = new GlideAggregate('fixed_asset');
totalCostSum.addAggregate('SUM', 'total_cost');
totalCostSum.query();
while (totalCostSum.next()) {
  var allTotalCost = 0;
  allTotalCost = totalCostSum.getAggregate('SUM', 'total_cost');
  aTotalCost = totalCostSum.getValue('total_cost');
  gs.print('Unique field value: ' + aTotalCost + ', SUM = ' + allTotalCost + ', ' + allTotalCost/aTotalCost + ' records');
}
```

The output for this example is:

*** Script: Unique field value: 12, SUM = 36, 3 records
*** Script: Unique field value: 10, SUM = 40, 4 records
*** Script: Unique field value: 5, SUM = 25, 5 records

Using the same data points as the prior example, if you use the groupBy() method, the SUM aggregate returns the sum of all values for the specified field.
The following example illustrates implementing the SUM aggregate using the `groupBy()` method:

```javascript
var totalCostSum = new GlideAggregate('fixed_asset');
totalCostSum.addAggregate('SUM', 'total_cost');
totalCostSum.groupBy('total_cost');
totalCostSum.query();
if(totalCostSum.next()){  // in case there is no result
    var allTotalCost = 0;
    allTotalCost = totalCostSum.getAggregate('SUM', 'total_cost');
gs.print('SUM of total_cost: = ' + allTotalCost);
}
```

The output for this example is:

```plaintext
*** Script: SUM of total_cost: 101
```

**GlideRecord**

GlideRecord is a special Java class (`GlideRecord.java`) that can be used in JavaScript exactly as if it was a native JavaScript class.

GlideRecord:

- is used for database operations instead of writing SQL queries.
- is an object that contains zero or more records from one table. Another way to say this is that a GlideRecord is an ordered list.

A GlideRecord contains both records (rows) and fields (columns). The field names are the same as the underlying database column names. For additional information, refer to .

**Note:** Use of `gs.sql()` scripting syntax was discontinued in Geneva. Use standard `GlideRecord` syntax in its place.

**Using GlideRecordSecure**

GlideRecordSecure is a class inherited from GlideRecord that performs the same functions as GlideRecord, and also enforces ACLs.

**Non-writable fields**

Be aware that, when using GlideRecordSecure, non-writable fields are set to NULL when trying to write to the database. By default, `canCreate()` on the column is replaced with `canWrite()` on the column. If that returns false, the column value is set to NULL.

**Checking for NULL values**

If an element cannot be read because an ACL restricts access, a NULL value is created in memory for that record. With GlideRecord, you must explicitly check for any ACLs that might restrict read access to the record. To do so, an if statement such as the following is required to check if the record can be read:

```javascript
if ( !grs.canRead() ) continue;
```

With GlideRecordSecure, you do not need to explicitly check for read access using `canRead()`. Instead, you can use `next()` by itself to move to the next record. The following example provides a comparison between GlideRecord and GlideRecordSecure.

```javascript
var count  = 0;
```
var gr  = new GlideRecord('mytable');
gr. query();
while (gr. next()) {
    if (!gr. canRead()) continue;
    if (!gr. canWrite()) continue;
    if (!gr. val. canRead() || !gr. val. canWrite())
        gr. val = null;
    else
        gr. val = "val-" + gr. id;
    if (gr. update())
        count ++;
}

var count  = 0;
var grs  = new GlideRecordSecure('mytable');
grs. query();
while (grs. next()) {
    grs. val = "val-" + grs. id;
    if (grs. update())
        count ++;
}

**Examples**

These are two simple examples using GlideRecordSecure.

var att  = new GlideRecordSecure ('sys_attachment');
att. get('$[sys_attachment.sys_id]');
var sm  = GlideSecurityManager.get();
var checkMe  = 'record/sys_attachment/delete';
var canDelete  = sm.hasRightsTo(checkMe,att);
gs. log('canDelete: ' + canDelete);

var grs = new GlideRecordSecure('task_ci');
grs.addQuery();
grs.query();
var count  = grs. getRowCount();
if (count  > 0 ) {
    var allocation  = parseInt(10000/count) / 100;
    while (grs.next()) {
        grs.u_allocation = allocation;
        grs.update();
    }
}

**GlideSystem**
The GlideSystem API provides methods for retrieving information.

The GlideSystem (referred to by the variable name ‘gs’ in business rules) provides a number of convenient methods to get information about the system, the current logged in user, etc. For example, the method addInfoMessage() permits communication with the user.

    gs.addInfoMessage('Email address added for notification');

Many of the GlideSystem methods facilitate the easy inclusion of dates in query ranges and are most often used in filters and reporting.

For additional information, see .
**GlideDateTime**

The **GlideDateTime** class provides methods for performing operations on GlideDateTime objects, such as instantiating GlideDateTime objects or working with *glide_date_time* fields.

In addition to the instantiation methods described below, a GlideDateTime object can be instantiated from a *glide_date_time* field using the `getGlideObject()` method (for example, `var gdt = gr.my_datetime_field.getGlideObject();`).

Some methods use the Java Virtual Machine time zone when retrieving or modifying a date and time value. Using these methods may result in unexpected behavior. Use equivalent local time and UTC methods whenever possible.

For additional information, refer to .

**GlideDate and GlideDateTime examples**

The **GlideDate** and **GlideDateTime** APIs are used to manipulate date and time values.

**Note:** This functionality requires a knowledge of JavaScript.

For additional information, refer to API and API.

You can create a GlideDateTime object from a GlideDate object by passing in the GlideDate object as a parameter to the GlideDateTime constructor. By default, the GlideDateTime object is expressed in the internal format, *yyyy-MM-dd HH:mm:ss* and the system time zone UTC.

```javascript
var gDate = new GlideDate();
gDate.setValue('2015-01-01');
gs.info(gDate);

var gDT = new GlideDateTime(gDate);
gs.info(gDT);
```

**Output:**

```
2015-01-01
2015-01-01 00:00:00
```

**JavaScript tools script includes reference**

Several script includes are available for working with JavaScript.

<table>
<thead>
<tr>
<th>Script Include</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
<td>Include class to pull in prototype server if referenced.</td>
</tr>
<tr>
<td><strong>GenericException</strong></td>
<td>Provides generic exception wrapping.</td>
</tr>
<tr>
<td><strong>GSLog</strong></td>
<td>Implements levels of log output to simplify logging and debugging from JavaScript.</td>
</tr>
<tr>
<td><strong>IllegalArgumentException</strong></td>
<td>Provides exception wrapping for a specified illegal argument. This class extends GenericException.</td>
</tr>
<tr>
<td><strong>isJavaObject</strong></td>
<td>Determines if a given value is an instance of a JavaScript object.</td>
</tr>
<tr>
<td><strong>JSONParser</strong></td>
<td>Parses JSON without using JavaScript.</td>
</tr>
<tr>
<td><strong>JSValidator</strong></td>
<td>Provides functions for validating JavaScript code.</td>
</tr>
</tbody>
</table>
**GenericException**
This JavaScript method provides generic exception wrapping.

**Where to use**

Use these methods to define exception handling in your JavaScript code.

**Method summary**

<table>
<thead>
<tr>
<th>Method summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialize(errorMessage)</td>
<td>Method called by the Prototype JavaScript Framework during object creation.</td>
</tr>
<tr>
<td>getMessage()</td>
<td>Gets the error message.</td>
</tr>
<tr>
<td>toString()</td>
<td>Converts the error message to a string.</td>
</tr>
</tbody>
</table>

**Method detail**

**initialize(errorMessage)**
Called by the Prototype JavaScript Framework during object creation to initialize a new instance of this class. Provide the parameter value but do not directly call this method.

- Input field parameters: `errorMessage` (string) the error message.
- Output field returns: void.

Example that returns void:

```
var jerr = new GenericException("This is an error");
```

**getMessage()**

Gets the error message.

- Input field parameter: none
- Output field returns: (string) the error message.

Example:

```
var jerr = new GenericException("This is an error");
gs.print(jerr.getMessage());
```

Returns the output: *** Script: This is an error

**toString()**

Converts the error message to a string.

- Input field parameter: none
- Output field returns: (string) the error message.
Example:

```javascript
var jerr = new GenericException("This is an error");
gs.print(jerr.toString());
```

Returns the output: *** Script: This is an error

`IsJavaObject`

Determines if a value is an instance of a Java object rather than a JavaScript primitive, JavaScript object, null, or undefined.

Where to use

Use in any JavaScript code if you need to test for a Java object instance.

Method summary

<table>
<thead>
<tr>
<th>Method summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isJavaObject(val)</code></td>
<td>Determines if a given value is an instance of a Java object.</td>
</tr>
</tbody>
</table>

Method detail

`isJavaObject(val)`

Determines if a given value is an instance of a Java object.

- Input field parameters: `val`: the value to test.
- Output field returns (boolean):
  - True if the given value is an instance Java object.
  - False if the value is a JavaScript primitive, JavaScript object, null, or undefined.

Example:

```javascript
var jsn = new JSON();
var jsary = [{ name: 'test', value: 1 }, { key: 'test2', value: 2}];
gs.print(isJavaObject(jsn));
```

Output returns: *** Script: false

`JSValidator`

Provides a helper function for validating JavaScript code, by getting any script errors generated by the Glide System. This class extends `AbstractAjaxProcessor`.

Where to use

Use this script include whenever you need to validate a script.
Method summary

<table>
<thead>
<tr>
<th>Method summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>validate()</td>
<td>Validates the JavaScript code.</td>
</tr>
</tbody>
</table>

Method detail

validate()

 Gets any script errors generated by the Glide System.

- Input field parameters: none
- Output field returns: Any script error messages resulting from the validation; if none have occurred, returns null.

Example:

```javascript
var jsv = new JSValidator();
gs.print(jsv.validate());
```

Schedule Pages

A schedule page is a record that contains a collection of scripts that allow for custom generation of a calendar or timeline display.

Creation of timeline schedule pages requires understanding of the page/event flow and the ability to write client and server side JavaScript.

Schedule pages form

To access schedule pages, navigate to System Scheduler > Schedules > Schedule Pages.

The form provides the following fields, depending upon the View Type selected:

<table>
<thead>
<tr>
<th>Field</th>
<th>Field Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>General name that is used to identify the current schedule page.</td>
</tr>
<tr>
<td>Field</td>
<td>Field Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Schedule type</td>
<td>String</td>
<td>The schedule type is a string that is used to uniquely identify the schedule page via the &quot;sysparm_page_schedule_type&quot; URI parameter. For example, a schedule page could be accessed as follows: &lt;br&gt; /show_schedule_page.do?sysparm_page_schedule_type=gantt_chart&amp;sysparm_timeline_task_id=d530bf907f0000015ce594fd929cf6a4 &lt;br&gt; Alternatively, the schedule page can also be accessed by setting the &quot;sysparm_page_sys_id&quot; URI parameter to that of the unique 32 character hexadecimal system identifier of the schedule page.</td>
</tr>
<tr>
<td>View Type</td>
<td>Choice</td>
<td>Each view type displays different field combinations. There are two options available: &lt;br&gt; - Calendars &lt;br&gt; - Timelines</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>General description that provides additional information about the current schedule page. This field is not necessary.</td>
</tr>
<tr>
<td>Init function name</td>
<td>String</td>
<td>The init function name specifies the name of the JavaScript function to call inside the Client script function for calendar type schedule pages.</td>
</tr>
<tr>
<td>HTML</td>
<td>String</td>
<td>The HTML field is a scriptable section that is parsed by Jelly and injected into the display page prior to the rest of the calendar. It can be used to pass in variables from the server and define extra fields if necessary.</td>
</tr>
<tr>
<td>Field</td>
<td>Field Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Client script</td>
<td>String</td>
<td>The client script is a scriptable section that allows for configuring options of the schedule page display. The API is different depending on the schedule page view type and is discussed below.</td>
</tr>
<tr>
<td>Server AJAX processor</td>
<td>String</td>
<td>Note: This functionality is only used by Calendar type schedule pages. The Server AJAX processor is specific to calendar type schedule pages that is used to return a set of schedule items and spans to be displayed.</td>
</tr>
</tbody>
</table>

**Timeline schedule pages**

A Timeline Schedule Page is a specific record that contains configuration information for displaying time based points and spans in a ‘timeline’ like fashion.

The timeline schedule page references a script include that extends from AbstractTimelineSchedulePage to perform dynamic modification to the timeline based on different events and conditions. Both the schedule page and the script include for timeline generation allow for extreme customization and their corresponding application programming interface (API) is documented below.

The following diagram shows the series of events that occur when a timeline schedule page is accessed. Once the timeline has been loaded, all subsequent events, such as events resulting from timeline interaction (e.g. moving a timeline span), follow the same logic flow shown in the gray event box.
Timeline Flow

Applications that use schedule pages to generate time lines

- Project Management
- Maintenance Schedules
- Group On-Call Rotation
- Field Service Management

Timeline schedule page example
The following example demonstrates how to create a timeline schedule page with corresponding script include utilizing a majority of the API described above.

For this example we are going to create an Incident Summary Timeline for a project support manager to visualize all of the new incidents. All new incidents should be displayed as single points where the priority of the incident is distinguished by a different point icon. Additionally, all
closed incidents should be displayed on the timeline in a separate group that shows the duration of the incident before it was closed. Since the Project Manager wants to be able to easily close new items that are resolved without using any form lists, we will handle the vertical move event allowing the new incidents to be dragged into the closed incident group or items within.

**Schedule Page**

Create a new schedule page with the following properties:

- **Name:** Hardware Incidents
- **Schedule type:** incident_timeline
- **View Type:** Timeline
- **Client Script:**

```javascript
// Set our page configuration
glideTimeline.setReadOnly(false);
glideTimeline.showLeftPane(true);
glideTimeline.showLeftPaneAsTree(true);
glideTimeline.showTimelineText(true);
glideTimeline.showDependencyLines(false);
glideTimeline.groupByParent(true);
glideTimeline.setDefaultPointIconClass('milestone');

// We will define what items to display and provide a custom event handler for moving new items to the closed state
getItems:

```javascript
getItems: function() {
    // Specify the page title
    this.setPageTitle('My Custom Incident Summary Timeline');

    var groupNew = new GlideTimelineItem('new');
groupNew.setLeftLabelText('New Incidents');
groupNew.setImage('../images/icons/all.gif');
groupNew.setTextBold(true);
this.add(groupNew);
```

**Script Include**

Now that the schedule page has been created we need to generate a matching script include for the events that were registered. Create a new script include with the following properties:

- **Name:** IncidentTimelineScriptInclude
- **Active:** Checked
- **Client callable:** Checked
- **Script:**

```javascript
// Class Imports
var IncidentTimelineScriptInclude = Class.create();
IncidentTimelineScriptInclude.prototype =
Object.extendsObject(AbstractTimelineSchedulePage, {  

```

```javascript
// GET_ITEMS
getItems: function() {  
    // Specify the page title
    this.setPageTitle('My Custom Incident Summary Timeline');

    var groupNew = new GlideTimelineItem('new');
groupNew.setLeftLabelText('New Incidents');
groupNew.setImage('../images/icons/all.gif');
groupNew.setTextBold(true);
this.add(groupNew);
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
var groupClosed = new GlideTimelineItem('closed');
groupClosed.setLeftLabelText('Closed Incidents');
groupClosed.setImage('../images/icons/all.gifx');
groupClosed.setTextBold(true);
groupClosed.setIsDroppable(true);

// This allows us to drag an open incident onto the closed group row.
this.add(groupClosed);

// Get all the incidents and let's add only the new/closed ones appropriately
var gr = new GlideRecord('incident');
gr.query();
while(gr.next()) {
    // Only loop through new/closed incidents
    if(gr.incident_state != '1' && gr.incident_state != '7') continue;
    // Ok, we have a new/closed incident. Create the item and the span first.
    var item = new GlideTimelineItem(gr.getTableName(), gr.sys_id);
    var span = item.createTimelineSpan(gr.getTableName(), gr.sys_id);

    // Specific properties for a new incident
    if(gr.incident_state == '1') { // New
        item.setParent(groupNew.getSysId());
        item.setImage('../images/icons/open.gifx');
        span.setTimeSpan(gr.getElement('opened_at').getGlideObject().getNumericValue(),
                         gr.getElement('opened_at').getGlideObject().getNumericValue());
        switch(gr.getElement('priority').toString()) {
        case '1': span.setPointIconClass('red_circle'); break;
        case '2': span.setPointIconClass('red_square'); break;
        case '3': span.setPointIconClass('blue_circle'); break;
        case '4': span.setPointIconClass('blue_square'); break;
        case '5': span.setPointIconClass('sepia_circle'); break;
        default: // Otherwise, the default point icon class will be used
        (Milestone)
    }
    } else if(gr.incident_state == '7') { // Closed
        item.setParent(groupClosed.getSysId());
        item.setImage('../images/icons/closed.gifx');
        span.setTimeSpan(gr.getElement('opened_at').getGlideObject().getNumericValue(),
                         gr.getElement('closed_at').getGlideObject().getNumericValue());
    }
    // Common item properties
    item.setLeftLabelText(gr.short_description);
    // Common span properties
    span.setSpanText(gr.short_description);
    span.setToolTip('<strong>' + GlideStringUtil.escapeHTML(gr.short_description) + '</strong><br>' +
                     gr.number);
    span.setAllowXMove(false);
    span.setAllowYMove(gr.canWrite() ? true:false);
    span.setAllowYMovePredecessor(false);
span.setAllowXDragLeft(false);
span.setAllowXDragRight(false);

// Now we add the actual item
this.add(item);
} } ,

moduleId: function(spanId, moduleId) {

  // Get information about the current incident
  var gr = new GlideRecord('incident');
  gr.addQuery('sys_id', spanId);
  gr.query();
  if(!gr.next())
    return this.setStatusError('Error', 'Unable to lookup the current incident.');

  // Get information about the dropped GlideTimelineItem. If it was dropped in an item on the 'New Incidents' group let's do nothing. If it was dropped in the 'Closed Incidents' group let's adjust the state automatically.
  var grDropped = new GlideRecord('incident');
  grDropped.addQuery('sys_id', moduleId);
  grDropped.query();
  if(!grDropped.next() || grDropped.incident_state == '7') {
    // This means the dropped item was either the 'Closed Incidents' group (which has no record or sys_id) or an existing incident that is closed. The 'New Incidents' also has no sys_id; however, the default behavior for items without a sysId is to be non-droppable. This is why we explicitly denoted the 'Closed Incidents' to be marked as "droppable".

    // Return a dialog prompt
    this.setStatusPrompt('Confirm', 'Are you sure you want to close: ' +
      '<div style="margin:10px 0 10px 14px;padding:4px;background-color:#EBEBEB;">
        <strong>' + GlideStringUtil.escapeHTML(gr.short_description) + '</strong><br/><div class="font_smaller">
      ' + gr.number + '</div>
    </div></div>',
      'this._elementMoveYHandler_DoClose', // This function is for when the OK button is clicked.
      'this._elementMoveYHandler_DoNothing', // This function is for when the Cancel button is clicked.
      'this._elementMoveYHandler_DoNothing'); // This function is for when the Close button is clicked.
  } }

  _elementMoveYHandler_DoClose: function(spanId, moduleId, newItemId) {

  // Get information about the current incident
  var gr = new GlideRecord('incident');
  gr.addQuery('sys_id', spanId);
  gr.query();
  if(!gr.next())
    return this.setStatusError('Error', 'Unable to lookup the current incident.');

  // Only allow the new incidents to have their state adjusted.
  if(gr.incident_state != '1')
    return this.setStatusError('Error','Only new incidents can have their state adjusted.');

  // Get information about the dropped GlideTimelineItem. If it was dropped in an item on the "New Incidents" group let's do nothing. If it was dropped in the "Closed Incidents" group let's adjust the state automatically.
  var grDropped = new GlideRecord('incident');
  grDropped.addQuery('sys_id', newItemId );
  grDropped.query();
  if(!grDropped.next() || grDropped.incident_state == '7') {
    // This means the dropped item was either the 'Closed Incidents' group (which has no record or sys_id) or an existing incident that is closed. The 'New Incidents' also has no sys_id; however, the default behavior for items without a sysId is to be non-droppable. This is why we explicitly denoted the 'Closed Incidents' to be marked as "droppable".

    // Return a dialog prompt
    this.setStatusPrompt('Confirm', 'Are you sure you want to close: ' +
      '<div style="margin:10px 0 10px 14px;padding:4px;background-color:#EBEBEB;">
        <strong>' + GlideStringUtil.escapeHTML(gr.short_description) + '</strong><br/><div class="font_smaller">
      ' + gr.number + '</div>
    </div></div>',
      'this._elementMoveYHandler_DoClose', // This function is for when the OK button is clicked.
      'this._elementMoveYHandler_DoNothing', // This function is for when the Cancel button is clicked.
      'this._elementMoveYHandler_DoNothing'); // This function is for when the Close button is clicked.
  } },
// Notice that this function takes the same function arguments as the original function for which it is a custom event handler for.

// Update the database record from 'New' to 'Closed'.
var gr = new GlideRecord('incident');
gr.addQuery('sys_id', spanId);
gr.query();
gr.next();
gr.setValue('incident_state', '7');
gr.update();

// This will re-render the timeline showing the updated item in the closed group.
this.setDoReRenderTimeline(true);

// Let's show a success message
this.setStatusSuccess('Success', '<strong>' + gr.short_description + '</strong> was successfully closed.');

// Since the user clicked cancel or close we simply do nothing.
_elementMoveYHandler_DoNothing: function(spanId, itemId, newItemId) { };

Screenshots / Results

1. Navigate to:
   http://[YOURINSTANCE]:8080/show_schedule_page.do?
sysparm_page_schedule_type=incident_timeline

   Notice the bold text is the value of the schedule page Schedule type field.

2. The page displays a timeline as specified by the schedule page and script include created. A link to this page can be created and placed as a module or UI action as necessary.
3. Attempting to move a closed incident anywhere displays the expected error message.

![Timeline Example Error Moving](image)

4. Moving the incident: **I need more memory** displays the following confirmation box.

![Timeline Example Confirm Close](image)

5. Clicking the **Cancel** button closes the overlay. Clicking the **OK** button actually updates the incident_state of the record and then displays the following success box.

![Timeline Example Close Success](image)

6. After clicking **OK**, it is clear the incident is now listed in the **Closed Incidents** group.
Using discovery script includes

Discovery script includes define JavaScript classes that you can use to accomplish Discovery tasks.

Using GlideRecordUtil to Work with GlideRecords

GlideRecordUtil is a utility class that provides methods that are useful for working with GlideRecords during Discovery.

Getting a GlideRecord Instance

To get a GlideRecord instance for a given configuration item, and of the correct class and table, use the getCIGR(sys_id) method. For example, the following code gets the GlideRecord of a CI with the sys_id of 2dfd7c8437201000deeabfc8bcbe5d56:

```javascript
var gr = new GlideRecordUtil().getCIGR("2dfd7c8437201000deeabfc8bcbe5d56");
```

To retrieve any hierarchical table without knowing its class type, use the getGR(base_table, sys_id) method. For instance, if you need to get a GlideRecord for a computer class CI, you may not know if it is a computer class or more specifically a Windows server class or Linux server class. Using this method guarantees that you have a GlideRecord with the correct class. This is important because different classes have different attributes—in this case, a Windows server has attributes different from those of a Linux server. You must get a GlideRecord in the correct class or attributes may be missing. The following is a typical example of how to do this:

```javascript
var gr = new GlideRecordUtil().getGR("cmdb_ci_computer", "2dfd7c8437201000deeabfc8bcbe5d56");
```

Getting All the Fields In a GlideRecord

The getFields(gr) method returns a JavaScript object, such as a hash map, of all the fields or attributes that exist in a given GlideRecord.

```javascript
var gr = new GlideRecordUtil().getGR("cmdb_ci_computer", "2dfd7c8437201000deeabfc8bcbe5d56");
var fields = new GlideRecordUtil().getFields(gr);
gs.log(fields.join(" ")); // List all the fields that are in a computer CI
```

Populating GlideRecord Object Fields

The populateFromGR(hashmap, gr, ignore) method allows you to take a GlideRecord object and populate its fields and values into a JavaScript object. The third argument (ignore) is an optional JavaScript object that allows you to exclude certain fields. For example, you may not care about sys_created_by or sys_updated_by fields in a GlideRecord.

```javascript
var objectToPopulate = { }
var gr = new GlideRecordUtil().getGR("cmdb_ci_computer", "2dfd7c8437201000deeabfc8bcbe5d56");
var ignore = {"sys_created_on": true, "sys_updated_by": true};
ngr.GlideRecordUtil().populateFromGR(objectToPopulate, gr, ignore);
```
The `mergeToGR(hashmap, gr, ignore)` method allows you to populate a GlideRecord with a field/value-paired object. The ignore argument stops specified fields from being updated. The following code example updates a computer record's `name` and `os` fields, but does not update the `sys_created_by` field:

```javascript
var gr = new GlideRecordUtil().getGR("cmdb_ci_computer", "2dfd7c8437201000deeabfc8bcbe5d56");
var obj = {"name": "xyz", "os": "windows 2000", "sys_created_by": "aleck.lin"};
var ignore = {"sys_created_by": true};
new GlideRecordUtil().mergeToGR(obj, gr, ignore);
gr.update();
```

### Getting Table Hierarchies

The `getTables(table)` method returns a list of table hierarchies, as shown in the following example:

```javascript
var tables = new GlideRecordUtil().getTables("cmdb_ci_linux_server");
gs.log(tables.join(",
"));
// The result would be "cmdb_ci, cmdb_ci_computer, cmdb_ci_server, cmdb_ci_linux_server".
```

### Using DiscoveryException and AutomationException

When writing Discovery sensors and sensor-related scripts, you may want to use `DiscoveryException` or `AutomationException` to indicate that an exception has come from Discovery.

The `DiscoveryException` script include extends `AutomationException`, which extends the `GenericException` class. The following example uses `DiscoveryException` to throw an exception:

```javascript
function foo() {
    if (//condition matches) throw new DiscoveryException("The message", "The cause");
}
```

The first argument takes the message of the exception and the second argument (optional) takes the cause of the exception. You may also want to catch the exception and log it as shown in the example below:

```javascript
try {
    foo();
} catch (e) {
    if (e instanceof DiscoveryException)
        gs.log("A DiscoveryException occurred. It is "+ e.getMessage()+ " caused by "+ e.getCause());
}
```

The above example also applies for `AutomationException`. `DiscoveryException` is typically used to provide exception processing specifically for Discovery, while `AutomationException` is used for exception processing that applies to both Orchestration and Discovery.

### vCenter API User Privileges

To determine the user privileges required by the VMware activity that logs into vCenter:
Determining Privileges

1. Navigate to the VMware API documentation.
2. In the vCenter API page, select **All Types** in the left navigation menu.

3. Click **U-Z** in the API index that appears.

4. Select **VirtualMachine** in the index. Information appears about a VirtualMachine object in the API.
5. Select **Local Methods** from the navigation menu at the top of the page.
XMLDocument script object

A JavaScript object wrapper for parsing and extracting XML data from an XML document (String).
Use this Javascript class to instantiate an object from an XML string, usually a return value from a Web Service invocation, or the XML payload of ECC Queue. Using the XMLDocument object in a Javascript business rule lets you query values from the XML elements and attributes directly.

**Programming interface**

**Constructor**

The constructor of a script object creates a new instance of the object to be used.

```javascript
var xmlString = "<test>
  <one>
    <two att="xxx">abcd1234</two>
    <three boo="yah" att="yyy">1234abcd</three>
    <two>another</two>
  </one>
  <number>1234</number>
</test>";
var xmldoc = new XMLDocument(xmlString);
```

To perform XML parsing of an XML string that is name space qualified, specify ‘true’ for the second argument for the XMLDocument constructor. The following is an example of parsing and XML string that contains name space qualification of its elements.

```javascript
var xmlString = "<bk:book xmlns:bk='urn:loc.gov:books'
  <bk:title>Cheaper by the Dozen</bk:title>
  <isbn:number>1568491379</isbn:number>
</bk:book>";
var xmldoc = new XMLDocument(xmlString, true); // XML document is name space aware
```

**Locating nodes and elements**

Now that we have the XMLDocument object, we can call the following APIs to locate nodes and elements of the XML document, as well as extract text from it directly. The query syntax for locating nodes and attributes is based on XPath.

**Note:** XML parsing with this object is not namespace aware, this means that if you are locating a node name with namespace in it eg. ‘<names:myelement> ...’, the XPath search string would be ‘//myelement’

The following are examples of locating a node by its XPath and getting the text value out of the resulting node.

```javascript
var str = xmldoc.getNodeText("//two"); // returns the first occurrence of the node
// str == "abcd1234"

str = xmldoc.getNodeText("//three");
// str == "1234abcd"

str = xmldoc.getNodeText("//test/one/*");
```
// str == "abcd1234"

str = xmldoc.getNodeInt("//number");
// str == 1234

The following examples locates the node by XPath and uses the API on node and element to get attributes and value.

var node = xmldoc.getNode("/test/one/two");
// node.getNodeName() == "two"
// node.getNodeType() == "1" // 1 == ELEMENT_NODE
// node.getLastChild().getNodeType() == "3" // 3 == TEXT_NODE
// node.getLastChild().getNodeValue() == "abcd1234"

Or you can use the following convenience functions to get the node name and type

str = xmldoc.getNodeName("//three");
// str == "three"

str = xmldoc.getNodeType("//three");
// str == "1"

You can also get a list of nodes that you can iterate or access directly by position

var nodelist = xmldoc.getNodes("//one/*"); // two, three, and two
// nodelist.getLength() == "3"
// nodelist.item(0).getNodeName() == "two"
// nodelist.item(1).getNodeName() == "three"

The following is an example of parsing an XML string that is qualified with name spaces.

var xmlString = "<bk:book xmlns:bk='urn:loc.gov:books'
xmlns:isbn='urn:ISBN:0-395-36341-6'>" +
"<bk:title>Cheaper by the Dozen</bk:title>" +
"<isbn:number>1568491379</isbn:number>" +
"</bk:book>";

var xmldoc = new XMLDocument(xmlString, true);
var str = xmldoc.getNodeText("//bk:title"); // returns the first occurrence
of the node
gs.log(str);

str = xmldoc.getNodeText("//bk:book/*");
gs.log(str);

str = xmldoc.getNodeInt("//isbn:number");
gs.log(str);

Getting attribute values

An attribute is just an extension of a node and so it has all the same APIs.
The following example shows how to query for a node to get its attribute by position

node = xmldoc.getNode("//two");
// node.getAttributes().item(0).getNodeValue() == "xxx"

str = xmldoc.getAttribute("//two", "att");
// str == "xxx"
XPath also has a query syntax for locating the attribute node directly, for example

```javascript
str = xmldoc.getNodeText("//*[@att="yyy"]");
// str == "1234abcd"

str = xmldoc.getNode("//@boo").getNodeValue();
// str == "yah"
```

**Creating new elements**

An XML element can be created at any level of the XML document once it has been created. Use the `setCurrent` function to position where the new element will be created as a child element, and use the `createElement` function to create the element.

```javascript
var xmlString = "<test>
    <one>
        <two att="xxx">abcd1234</two>
        <three boo="yah" att="yyy">1234abcd</three>
        <two>another</two>
    </one>
    <number>1234</number>
</test>";

var xmldoc = new XMLDocument(xmlString);
xmldoc.createElement("new", "test"); // creates the new element at the
document element level if setCurrent is never called
xmldoc.createElement("new2"); // calling without a value will create a new
element by itself

var el = xmldoc.createElement("new3");
xmldoc.setCurrent(el); // this is now the parent of any new elements created
subsequently using createElement()
xmldoc.createElement("newChild", "test");
```

The resulting XML document looks like this

```xml
<test>
    <one>
        <two att="xxx">abcd1234</two>
        <three boo="yah" att="yyy">1234abcd</three>
        <two>another</two>
    </one>
    <number>1234</number>
    <new>test<new>
        <new2/>
        <new3>
            <newChild>test</newChild>
        </new3>
    </new>
</test>
```

**XMLHelper**

The XML helper script include makes it easy to parse XML in scripts.

The script include converts your XML document into a JavaScript object.

The following changes were made to the XML helper script include:

- The `toObject()` method returns an object whose properties are all JavaScript objects. This method works properly whether the supplied parameter is an XML document or an XML string.
The methods `toXMLDoc()` and `toXMLStr()` have been made available. These methods are the inverse of the existing `toObject()` method.

The `toObject()` method has been extended to take an optional parameter of the XML input to convert as an alternative to the (still present) mechanism of specifying the XML input in the constructor.

**Note:** You must escape ampersand characters (&amp) in your XML or the conversion silently fails.

### Example

The following example takes an example XML document and then converts it into a JavaScript object. It then takes the outputted JavaScript object and uses a recursive function to output all members of the object. The recursive function is useful and reusable if you have any questions about how a particular XML document will be structured after being converted to a JavaScript object.

#### Script

```javascript
var xmlString = "<company>
  <employee id="10">
    <id>10</id>
    <firstname>Tom</firstname>
    <lastname>Cruise</lastname>
    <test>test1</test>
    <test>test3</test>
  </employee>
  <employee id="20">
    <id>20</id>
    <firstname>Paul</firstname>
    <lastname>Enderson</lastname>
    <test>test6</test>
    <test>test5</test>
  </employee>
  <employee id="30">
    <id>30</id>
    <firstname>Paul</firstname>
    <lastname>Bush</lastname>
    <test>test2</test>
    <test>test4</test>
  </employee>
</company>";

var helper = new XMLHelper(xmlString);
var obj = helper.toObject();

function logObj(obj, sep) {
  for (x in obj) {
    if (typeof obj[x] != "function") {
      gs.log(sep + x + ':' + obj[x]);
      logObj(obj[x], sep + '*');
    }
  }
}

logObj(obj, "*");
```

#### Output

*** Script: *employee:: [object Object], [object Object], [object Object]*
*** Script: **2:: [object Object]***
*** Script: ****id:: 30***
*** Script: ****test:: test2, test4***
*** Script: ****0:: test2***
*** Script: ****1:: test4***
*** Script: ***firstname:: Paul***
*** Script: ***lastname:: Bush***
*** Script: **0:: [object Object]***
*** Script: ****id:: 10***
*** Script: ****test:: test1, test3***
*** Script: ****0:: test1***
*** Script: ****1:: test3***
*** Script: ***firstname:: Tom***
Useful scripts

Scripts that provide useful functionality not included in the core system.

Get a user object

In a business rule or other server script, the `gs.getUser()` method returns a user object. The user object is an internal representation of the currently logged in user and provides information about the user and various utility functions.

For a list and description of the available scoped methods for the user object, see .

1. Retrieve the current user.

```javascript
var myUserObject = gs.getUser();
```

2. Use the `getUserByID` method to fetch a different user using the `user_name` field or `sys_id` on the target record.

For example:

```javascript
var ourUser = gs.getUser();
gs.print(ourUser.getFirstName()); //print the first name of the user you are currently logged in as
newUser = ourUser.getUserByID('-abel.tuter'); //fetch a different user, using the user_name field or sys_id on the target user record.
gs.print(newUser.getFirstName()); //first name of the user you fetched above
gs.print(newUser.isMemberOf('-Capacity Mgmt'));
```

Accessing the workflow scratchpad from business rules

A catalog item has been requested, the attached workflow contains a run script activity that populates a value in the scratchpad. From a business rule running on the requested item, we want to retrieve or set scratchpad values.

Prerequisites

Role required: admin

Name: Access Workflow Scratchpad from Business Rules

Type: Business Rule

Table: sc_req_item (Requested Item)

Description: A catalog item has been requested, the attached workflow contains a run script activity that populates a value in the scratchpad. From a business rule running on the requested item, we want to retrieve or set scratchpad values.

Parameters: n/a
Script:

```javascript
// the run script activity sets a value in the scratchpad
workflow.scratchpad.important_msg = "scratch me";

// get the workflow script include helper
var workflow = new Workflow();

// get the requested items workflow context
// this will get all contexts so you'll need to get the proper one if you
// have multiple workflows for a record
var context = workflow.getContexts(current);

// make sure we have a valid context
if (context.next()) {
  // get a value from the scratchpad
  var msg = context.scratchpad.important_msg;
  // msg now equals "scratch me", that was set in the run script activity

  // add or modify a scratchpad value
  context.scratchpad.status = "completed";

  // we need to save the context record to save the scratchpad
  context.update();
}
```

Add a field to the service catalog checkout

This is an example of adding a **Company** field to the checkout below the **Requested for** field using non-cart layout macros, that is, `glide.sc.use_cart_layouts` is false.

This field will then passing the value of that field to the **Company** field of the Service Catalog Request.

This example makes the following assumptions.

- This example is for an instance using two-step checkout. If two-step checkout is not enabled, enable it before beginning.
- This example populates the **Company** field on the Service Catalog Request form. If the field does not appear on the form, configure the form before beginning.

1. Go to **System UI > UI Macros** and select **servicecatalog_cart_template**.
2. Find where there are hidden variables declared and add the following line:

   ```html
   <input type="HIDDEN" name="cart_id" id="cart_id" value="\$[sc_cart.sys_id]\" />
   ```

3. Find the following code, which generates the **Requested For** code:

   ```html
   <tr class="header">
   ```
<td width = "30%">
${gs.getMessage('Requested for')}:
</td>
<td width="70%">
<label for="requestor_location">${gs.getMessage('Deliver to')}:
</label>
</td>
</tr>
<tr><td>$[SP]</td></tr>
<tr><td valign="top">
<j2:if test="$[jvar_can_delta_rf == false]">
$[sc_cart.requested_for.getDisplayValue()]
</j2:if>
<j2:if test="$[jvar_can_delta_rf != false]">
<g2:catalog_requested_for />
</j2:if>
</td>
<td>
<textarea id="requestor_location" style="width:100%" rows="4"
name="requestor_location" wrap="soft"
onChange="catDeliveryAddress('$[sc_cart.sys_id]',
'requestor_location');">
$[sc_cart.delivery_address]
</textarea>
</td>
</tr>
<tr><td>$[SP]</td></tr>
</tr>
</table>

4. Add the following code afterwards:

```
<tr class="header">
  <td colspan="2">Company</td>
</tr>
<tr><td>$[SP]</td></tr>
<tr><td colspan="2">
<g2:ui_reference name="core_company" table="core_company"
onchange="setCartValue()"/>
</td>
</tr>
<tr><td>$[SP]</td></tr>
</table>
```

- Note that 'ui_reference' is another macro that defines a reference field. There are several macros for different field types. You can see examples of these field types under System UI -> UI Macros. These macros start with 'ui_'. For this example, the reference field created is named core_company.

5. Now navigate to System UI > UI Pages and select the servicecatalog_checkout_one UI Page. Add this script to the Client script field:

```javascript
function setCartValue() {
  var newField = gel('core_company');
  var myCart = gel('cart_id');
  var cart_item = new GlideRecord('sc_cart_item');
  cart_item.addQuery('cart', myCart.value);
  cart_item.query();
}
```
if(cart_item.next()) {
    cart_item.hints = "<hints><entry key='sysparm_processing_hint'
value='setfield:request.company=" + newField.value + "]'</entry></hints>">
    cart_item.update();
}

For this example, the reference field was called **core_company**, and the field being populated on the request is **company**. If different fields are used:

- Find this line: `var company = gel('core_company');` and replace **core_company** with the name of the field in the checkout.
- In the line that starts with `cart_item.hints` replace `request.company` with the name of the field to be populated on the request ticket where `request` is the request being generated and `company` is the name of the field.

Now, when an item is ordered, the company field appears on the Catalog form:

![Shopping Cart Image]

**Add role to every user**

Adds a role to every user.
Prerequisites

Note: This functionality requires a knowledge of JavaScript.

Role required: admin

In this sample, the role being added is Self Service. To add a different role, simply substitute the desired role for self_service.

Name: Add Role to Every User
Type: Client Script, Background Script
Table: sys_user
Description: Adds a role to every user. In this sample, the role being added is Self Service. To add a different role, simply substitute the desired role for self_service.
Parameters:

Script:

```javascript
var gr = new GlideRecord("sys_user");
gr.query();
while (gr.next()) { 
  if (gr.accumulated_roles.toString().indexOf","self_service,") == -1) { 
    gr.roles = gr.roles + ",self_service";
    gr.update();
  }
}
```

Assign a catalog item to a group based on a delivery plan task

This assignment rule assigns a service catalog item to the database group if it uses a delivery plan that has a catalog task assigned to the desktop group.

Prerequisites

Role required: admin

Caution: The customization described here was developed for use in specific instances, and is not supported by ServiceNow Technical Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

Name: Assign Catalog Item to Group Based on Delivery Plan Task
Type: Assignment Rule
Table:
Description: This assignment rule assigns a service catalog item to the database group if it uses a delivery plan that has a catalog task assigned to the desktop group.
Parameters:

Script:

```javascript
//Return catalog items that have no group but do have a delivery plan assigned var ri  = new GlideRecord ( "sc_cat_item" ) ;
ri.addQuery("group", "=", null);
ri.addQuery("delivery_plan", "!=" , null);
```
ri.query();
while(ri.next()) {
    gs.log("Found an item");
    //Return tasks that point to the same delivery plan as the above item
    var dptask = new GlideRecord("sc_cat_item_delivery_task");
    dptask.addQuery("delivery_plan", "," , ri. delivery_plan);
    dptask.query();
    while(dptask.next()) {
        gs.log("Found a task");
        var gp = dptask.group.getDisplayValue();
        gs.log(gp);
        //If the task is assigned to desktop, assign the item's group to desktop
        if (dptask.group.getDisplayValue() == "Desktop") {
            ri.group.setDisplayValue("Desktop");
            gs.log("updating "+ ri.getDisplayValue());
            ri.update();
            break; }
    }
}

Change form color on state change

Changes color of a form field of the form on state change. The script can easily be changed to adjust any property of any object on the page accessible via the HTML DOM.

Caution: The customization described here was developed for use in specific instances, and is not supported by ServiceNow Technical Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

Name: Change Form Color on State Change
Type: Client Script
Table:
Description: Changes color of a form field of the form on state change. The script can easily be changed to adjust any property of any object on the page accessible via the HTML DOM.
Parameters:
Script:

```javascript
function onChange(control, oldValue, newValue, isLoading) {
    var elementID = gel("incident.priority");
    switch(newValue) {
        case "1": elementID.style.backgroundColor = "red"]; break;
        case "2": elementID.style.backgroundColor = "tomato"]; break;
        case "3": elementID.style.backgroundColor = "orange"]; break;
        case "4": elementID.style.backgroundColor = "yellow"]; break;
        case "5": elementID.style.backgroundColor = "green"]; break;
        default: elementID.style.backgroundColor = "white"; break;
    }
}
```

Create a UI routing action

This solution enables you to create a record with the service desk without knowing whether it is an incident or request item; the service desk can then route the record to the appropriate table.

Note: Functionality described here requires the Admin role.

To create a UI routing action:
1. Create a new table that extends the task table (for example, New Call).
2. Create a module to create a new New Call record.
3. Create any fields that you want on the New Call table.

The only fields you need are those fields necessary to determine whether the new call should route to an Incident or a Request Item. Ensure that the form contains any fields that you want to pass to the Incident or Request Item. In this example, the following are created on the form:

- **Requested for** (reference)
- **Location** (reference)
- **Call type** (choice with two values—Incident and Request)
- **Request Item** (reference to the sc_cat_item Item table)

4. Add some UI policies to set a couple of fields to mandatory and hide the Request Item field based on the Call type selection.
5. Remove unnecessary buttons and functionality from the form.
6. Create a new UI Action button. This button redirects the user to either an incident or a request. It also creates the incident record and copies values to the incident and the Request Item form.

```javascript
var reqItem = current.u_item;
var requestedFor = current.u_requested_for;
var location = current.location;

if(current.u_incident_request == 'Incident'){
    //Create a new incident record and redirect to the new incident
    var rec = new GlideRecord('incident');
    rec.initialize();
    rec.caller_id = requestedFor;
    rec.location = location;
    rec.insert();
    action.setRedirectURL(rec);
}

if(current.u_incident_request == 'Request'){
    //Build the url and route the user to the request item
    var url = ''; 
    if(current.u_item.sys_class_name == 'sc_cat_item_guide'){
        url = 'com.glideapp.servicecatalog_cat_item_guide_view.do?' +
        reqItem + '&sysparm_initial=true' +
        '&sysparm_guide=' +
        '&sysparm_user=' + requestedFor +
        '&sysparm_location=' + location;
    } else{
        url = 'com.glideapp.servicecatalog_cat_item_view.do?' +
        reqItem + '&sysparm_user=' +
        requestedFor +
        '&sysparm_location=' + location;
    }
    action.setRedirectURL(url);
}
```

7. The Route button in the preceding example passes the Requested for and Location values in the URL to the Request Item form. Ensure that you have variables called requested_for and location on your item, record producer, or order guide that map these values using the following client script. There is a limit as to how much information you can pass, as the URL has a restricted length. Avoid sending information from long text fields using this method.

```javascript
function onLoad() {
    var url = document.location.toString();
    var userKey = 'sysparm_user=';
    var locKey = 'sysparm_location=';
    var userPosition = url.indexOf(userKey);
    var locPosition = url.indexOf(locKey);
}```
Custom approval UI macro

This section describes how to create a custom approval UI macro.

**Caution:** The customization described here was developed for use in specific instances, and is not supported by ServiceNow Technical Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

**Script Name:** Custom Approval UI Macro

**Type:** UI Macro

**Description:** Here is an option to get more detail out of the My Approvals view of an Execution Plan. This can be done by creating a new UI macro. Navigate to System UI and click UI Macros. First, you will need to rename the existing `approval_summarizer_sc_task` to something like `approval_summarizer_sc_task_old` and deactivate it. Then, you will need to create a new one using the same name (`approval_summarizer_sc_task`). The name should basically tell you what the macro does and what it applies to. In this case, we're replacing an existing one so we decided to re-use the existing name.
Approval macros

Then you should copy the xml script at the bottom of this article into the xml code window in the new UI macro. This is great way to give some detail to an approver when you are doing line item approvals via approval tasks within the Service Catalog Execution Plans.

The old way

This is the view you see in My Approvals when using an approval task via the old method.
My Approvals (old way)

Notice there is not much detail telling the approver what they are actually approving. You can see the short description of the task but not much around what the item is.

The new way

This is the view you will see if you use the xml script below in place of the OOB (out-of-box) UI macro.

My Approvals
Using this method you can see details much like the request approval. You have a link into the item ordered, a short description (which contains the ability to expand the variables from the item), price, quantity and the total price. This helps the approver in that it shows more detail. They can now see what they are actually approving.

Script:

```xml
<?xml version="1.0" encoding="utf-8"?>
<j:jelly trim="true" xmlns:j="jelly:core" xmlns:g="glide"
xmlns:j2="null" xmlns:g2="null">
<tr>
<td class="label_left" width="100%">
<label style="margin-left: 10px">
${gs.getMessage('Summary of Item being approved')}: <input style="visibility: hidden" NAME="make_spacing_ok"></input>
</label>
</td>
</tr>
<g:evaluate var="jvar_ni" expression="
var sc_task = ${ref}.sysapproval;
var sc_req_labels = new GlideRecord('sc_req_item');
sc_req_labels.initialize();
var sc_req_item = new GlideRecord('sc_req_item');
sc_req_item.addQuery('request_item', sc_task.request_item.sys_id);
sc_req_item.query();
"/>
<tr width="100%">
<table width="100%">
<tr>
<td class="label_left" width="150px">
<label style="margin-left: 10px">
${sc_task.request_item.request.opened_by.sys_meta.label}:
</label>
</td>
<td>
${sc_task.request_item.request.opened_by.getDisplayValue()}
</td>
</tr>
<tr>
<td class="label_left" width="150px">
<label style="margin-left: 10px">
${sc_task.request_item.request.requested_for.sys_meta.label}:
</label>
</td>
<td>
${sc_task.request_item.request.requested_for.getDisplayValue()}
</td>
</tr>
<tr>
<td class="label_left" width="150px">
<label style="margin-left: 10px">
${gs.getMessage('Total')}:
</label>
</td>
<td>
<g:currency_format
value="${sc_task.request_item.request.price}"/>
</td>
</tr>
</table>
</td>
</tr>
</j:jelly>
```
<table>
<thead>
<tr>
<th>${sc_req_labels.number.sys_meta.label}</th>
<th>${sc_req_labels.description.sys_meta.label}</th>
<th>${sc_req_labels.price.sys_meta.label}</th>
<th>${sc_req_labels.quantity.sys_meta.label}</th>
<th>${gs.getMessage('Total')}</th>
</tr>
</thead>
<tbody>
<tr>
<td>${sc_task.request_item.number}</td>
<td>${sc_task.request_item.cat_item.short_description}</td>
<td>${sc_task.request_item.price * sc_task.request_item.quantity}</td>
<td>${sc_task.request_item.quantity}</td>
<td>${jvar_overall_total + jvar_item_price}</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Display field messages

Rather than use JavaScript `alert()`, for a cleaner look, you can display an error on the form itself. The methods `showFieldMsg()` and `hideFieldMsg()` can be used to display a message just below the field itself.

`showFieldMsg` and `hideFieldMsg` are methods that can be used with the `g_form` object.

These methods are used to change the form view of records (Incident, Problem, and Change forms). These methods may also be available in other client scripts, but must be tested to determine whether they work as expected.

When a field message is displayed on a form on load, the form scrolls to ensure that the field message is visible. Ensuring that users do not miss a field message because it was off the screen.

The global property `glide.ui.scroll_to_message_field` controls automatic message scrolling when the form field is offscreen (scrolls the form to the control or field).
## Method Detail

### showFieldMsg(input, message, type, (scrollform))
- **input** — name of the field or control
- **message** — message you would like to appear
- **type** — 'info', 'error', or 'warning'; defaults to info if not supplied
- **scroll form** — (optional) Set scrollForm to false to prevent scrolling to the field message offscreen

### Error Message

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
</table>
| g_form.showFieldMsg('impact','Low impact not allowed with High priority','error'); | Impact: 3-Low  
Low impact not allowed with High priority |

### Informational Message

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
</table>
| g_form.showFieldMsg('impact','Low impact response time can be one week','info'); | Impact: 3-Low  
Low impact response time can be one week |

### hideFieldMsg(input)
- **input** — name of the field or control
- **clearAll** — (optional) boolean parameter indicating whether to clear all messages. If true, all messages for the field are cleared. If false or empty, only the last message is removed.

### Removing a Message

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g_form.hideFieldMsg('impact');</td>
<td>//this will clear the last message printed to the field</td>
</tr>
</tbody>
</table>

## Legacy support

The showErrorBox() and hideErrorBox() are still available but simply call the new methods with type of error. You should use the new methods.

## GSLog

GSLog is a script include that simplifies script logging and debugging by implementing levels of log output, selectable by per-caller identified sys_properties values.
Log level

Logs can be at the level of Debug, Info, Notice, Warning, Err, or Crit (after BSD syslog.h and followers). The default logging level is Notice, so levels should be chosen accordingly.

Where to use

Use for any server-side script where you want to implement event logging.

Available Methods

initialize(traceProperty, caller)

Called by the Prototype JavaScript Framework during object creation to initialize a new instance of this class. Provide the input parameters, but do not call this method directly.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>traceProperty</td>
<td>String</td>
<td>System property that contains a value indicating the level at or above which messages will be written to the log.</td>
</tr>
<tr>
<td>caller</td>
<td>String</td>
<td>Name of the script calling the logger.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
```

logDebug(msg)

Logs debug events.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to write to the log</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.logDebug("This is a debug message");
```

Output:

```text
*** Script [TaskSLA]: This is a debug message
```

logInfo(msg)

Logs information events.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to write to the log</td>
</tr>
</tbody>
</table>

Example

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.logInfo("This is an info message");
```

logNotice(msg)

Logs notice events.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to write to the log</td>
</tr>
</tbody>
</table>

Example

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.logNotice("This is a notice message");
```
Example

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.logNotice("This is a notice");
```

Output:

```plaintext
*** Script [TaskSLA]: This is a notice
```

`logWarning(msg)`

Logs warning events.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to write to the log</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.logWarning("This is a warning message");
```

Output:

```plaintext
*** Script [TaskSLA]: This is a warning message
```

`logErr(msg)`

Logs error events.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to write to the log</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
Example

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.logErr("This is an error message");
```

Output:

```text
*** Script [TaskSLA]: This is an error message
```

$logCrit(msg)$

Logs critical events.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to write to the log</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.logCrit("This is a critical message");
```

Output:

```text
*** Script [TaskSLA]: This is a critical message
```

$logAlert(msg)$

Logs alert events.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to write to the log</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
Example

dev gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.logAlert("This is an alert");

Output:

*** Script [TaskSLA]: This is an alert

logEmerg(msg)
Logs emergency events.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to write to the log</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

Example

dev gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.logEmerg("This is an emergency message");

Output:

*** Script [TaskSLA ]: This is an emergency message

log(level, msg)
Logs a message at the specified level.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>level</td>
<td>String</td>
<td>Log level</td>
</tr>
<tr>
<td>msg</td>
<td>String</td>
<td>Message to write to the log</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
Example

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.log("debug", "debug message");
```

**getLevel(level)**

Returns the log level.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>level</td>
<td>String</td>
<td>Optional. Log level</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Log level</td>
</tr>
</tbody>
</table>

**Example**

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.setLevel("debug");
gs.print(gl.getLevel());
```

Output:

```text
*** Script: debug
```

**setLevel(level)**

Sets the log level.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>level</td>
<td>String</td>
<td>Log level to set</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.setLevel("debug");
```
debugOn()

Determines if debug is turned on.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>If true, debug is on; if false, debug is off.</td>
</tr>
</tbody>
</table>

Example

```javascript
var gl = new GSLog("com.snc.sla.tasksla.log", "TaskSLA");
gl.setLevel("debug");
gs.print(gl.debugOn());
```

Output:

```text
*** Script : true
```

Modify a GlideDateTime field value

This example demonstrates how to modify a GlideDateTime field value using a script.

**Name:** Modify a GlideDateTime Field Value

**Type:** A server side script that accesses a GlideDateTime field.

**Table:** N/A

**Description:** Given a GlideDateTime field or script object, show a variety of ways to easily modify value. The same concept also applies to the GlideDate object.

**Parameters:** N/A

**Script:**

```javascript
//You first need a GlideDateTime object
//this can be from instantiating a new object "var gdt = new GlideDateTime()"
//or getting the object from a GlideDateTime field
//getting the field value (for example: var gdt = current.start_date) only returns the string value, not the object
//to get the object use var gdt = current.start_date.getGlideObject();
//now gdt is a GlideDateTime object
var gdt = current.start_date.getGlideObject();

//All methods can use negative values to subtract intervals

//add 1 hour (60 mins * 60 secs)
gdt.addSeconds(3600);

//add 1 day
//gdt.addDaysLocalTime(1);
```
// subtract 1 day
gdt.addDaysLocalTime(-1);

// add 3 weeks
gdt.addWeeksLocalTime(3);

// subtract 6 months
gdt.addMonthsLocalTime(-6);

// add 1 year, representing the date and time using the UTC timezone instead of the local user's timezone.
gdt.addYearsUTC(1);

// set the value of the GlideDateTime object to the current session timezone/format
GlideSession.get().setTimeZoneName('US/Eastern');
gdt.setDisplayValue('2018-2-28 00:00:00');
gs.info('In ' + GlideSession.get().getTimeZoneName() + ': ' +
gdt.getDisplayValue());

Reprocess received emails

It is possible to reprocess emails that were received by your instance.

A similar "Reprocess received emails" UI action gives a list choice option at the bottom of the email list, which can be used to reprocess multiple emails at once.

A UI action named "Reprocess Email" puts a button on the email form:

Reprocess Email UI action

The script and condition are duplicated below for easy cutting and pasting. This UI action gives you a button when you open an email (in System Logs --> Email) that has a type of "received" or "received-ignored". It will create an event that will reprocess that incoming email through the Inbound Email Actions.
Condition

```javascript
current.type == 'received' || current.type == 'received-ignored'
```

Script

```javascript
var evt = new GlideRecord('sysevent');
evt.initialize();
evt.process_on = gs.nowDateTime();
evt.name = "email.read";
evt.parm1 = current.sys_id;
evt.insert();
gs.addInfoMessage('event created to reprocess email "' + current.subject + '"');
```

Sample ASP.NET with C Sharp redirect with cookies

This sample ASP.NET code creates a simple authentication portal and passes an unencrypted HTTP header as a cookie.

**Note:** Functionality described here requires the Admin role.

**Note:** Cookies are domain specific and cannot be used across different network domains. The only domain that can read a cookie is the domain that sets it. It does not matter what domain name you set. If you do not have an option of your SSO portal being in the same network domain as your ServiceNow instance (for example, in an on-premises deployment, an alternative is to pass the SSO token using URL GET or POST parameters.

This sample assumes:

- The web server supports ASP.NET and C#
- The target ServiceNow instance is `https://<instance name>.service-now.com/`
- SiteMinder or another single sign-on application has pre-authenticated the user
- The target ServiceNow instance expects an HTTP header of SM_USER

Change the ASP code to redirect users to the proper ServiceNow instance.

```html
<html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
        <title>Portal Page Login</title>
        <%--    <meta http-equiv="REFRESH" content="0;url=https://<instance name>.service-now.com/">--%>
    </head>
    <body>
        <form id="form1" runat="server">
            <h2><b>Portal Page Login</b></h2>
            <hr style="position: static" />
            <br />
            <asp:Label ID="Label2" runat="server" Font-Size="Larger" Height="21px" Style="position: static" Text="Instance URL:" Width="113px"></asp:Label>
            <asp:TextBox ID="urlBox" runat="server" Font-Size="Large" Style="position: static" Text=""></asp:TextBox><br />
            <br />
            <asp:Label ID="Label1" runat="server" Font-Size="Larger" Height="17px" Style="position: static" Text="User Id:" Width="113px"></asp:Label>
            <asp:TextBox ID="usernameBox" runat="server" Font-Size="Large" Style="position: static"></asp:TextBox><br />
            <br />
            <asp:Label ID="Label3" runat="server" Font-Size="Larger" Height="21px" Style="position: static" Text="Password:" Width="113px"></asp:Label>
            <asp:TextBox ID="passwordBox" runat="server" Font-Size="Large" Style="position: static"></asp:TextBox><br />
        </form>
    </body>
</html>
```
The following C# code handles the OnClick button event for the form. The code:

- Creates the cookie “SM_USER”
- Performs a redirect to the URL specified on the ASP form.

Change the C# code to create the proper cookie name.

```csharp
using System;
using System.Data;
using System.Configuration;
using System.Web;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void Button1_click(object sender, EventArgs e)
    {
    
```
### Useful approval assignment scripts

This is a searchable version of the useful approval and assignment scripts.

**Caution:** The customization described here was developed for use in specific instances, and is not supported by ServiceNow Technical Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community [forum](#).

For an easy-to-navigate version, visit the Useful Scripts portal.

#### Assign a group for ESS requests

**Type:** Assignment Rule

**Description:** This script automatically assigns a group for all ESS Requests.

**Script example:**

```java
try {
    HttpCookie myCookie = new HttpCookie("SM_USER");
    myCookie.Value = userNameBox.Text;
    Response.Cookies.Add(myCookie);
    Response.Redirect(urlBox.Text);
} catch {}
}
```

#### Assign Catalog Item to Group Based on Delivery Plan Task

**Type:** Assignment Rule

**Description:** This assignment rule assigns a service catalog item to the database group if it uses a delivery plan that has a catalog task assigned to the desktop group.

```java
//Return catalog items that have no group but do have a delivery plan assigned
var ri = new GlideRecord("sc_cat_item");
ri.addQuery("group", ",null");
ri.addQuery("delivery_plan", ",null");
ri.query();
while(ri.next()){
    gs.log("Found an item");
    //Return tasks that point to the same delivery plan as the above item
    var dptask = new GlideRecord("sc_cat_item_delivery_task");
    dptask.addQuery("delivery_plan", ri.delivery_plan);
    dptask.query();
    while(dptask.next()){
        gs.log("Found a task");
        var gp = dptask.group.getDisplayValue();
        gs.log(gp);
        //If the task is assigned to desktop, assign the item's group to desktop
        if(dptask.group.getDisplayValue() == "Desktop"){
            ri.group.setDisplayValue("Desktop");
            gs.log("updating "+ ri.getDisplayValue());
            ri.update();
            break;
        }
    }
}
```

#### Assign items with one task
Type: Assignment Rule
Description: Automatically assigns any catalog items with only one task associated to a particular group.

```javascript
//Get the catalog item for the current requested item
var scCatItem = new GlideRecord("sc_cat_item");
if(scCatItem.get('sys_id', current.cat_item)){
    // If the catalog item already has an assignment group or if using workflow
    we don't need to make an assignment
    if(!scCatItem.delivery_plan.nil()&& scCatItem.group.nil()){  
        var dpTask = new GlideRecord("sc_cat_item_delivery_task");
        dpTask.addQuery("delivery_plan","=",scCatItem.delivery_plan);
        dpTask.query();
        if(dpTask.getRowCount()==1&& dpTask.next()){  
            // Check that there is only 1 record in the GlideRecord
            dpTask.group;}}}
```

Assignment based on workload

Type: Business Rule
Description: Populate the assigned to based on the assignment group member who has the least amount of active incidents.

Parameters:
- order: >1000 if you want to execute after assignment rules
- condition: current.assigned_to == "" && current.assignment_group != ""
- when: before, insert/update

```javascript
var assignTo = getLowestUser();
gs.addInfoMessage("assigning to is " + assignTo);
current.assigned_to= assignTo;

function getLowestUser(){
    var userList = new Array();
    var cg = new GlideRecord('sys_user_grmember');
    cg.addQuery('group', current.assignment_group);
    cg.query();
    while(cg.next()){  
        var tech = cg.user.toString();
        var cnt = countTickets(tech);
        gs.addInfoMessage("Tech counts " + cg.user.name+' '+ cnt +" " + tech);
        userList.push({ sys_id: tech, name: cg.user.name, count: cnt });}

    for(var i=0; i < userList.length; i++){
        gs.addInfoMessage(userList[i].sys_id+" "+ userList[i].name+" " +
        userList[i].count);
        userList.sort(function(a, b){
            gs.addInfoMessage("Sorting: " + a.sys_id+"(" + a.count+"");
            " + b.sys_id+"(" + b.count+"))");
            return a.count- b.count;});

    if(userList.length<=0) return"";
    return userList[0].sys_id;}

function countTickets(tech){
    var ct = new GlideRecord('incident');
    ct.addQuery('assigned_to',tech);
    ct.addQuery('active',true);
    ct.query();
```
Run assignment rules when category is changed

Type: Client script

Table: Incident

Description: This example is an onChange client script on the category field within Incident. Note: this script used to use synchronous AJAX (asynchronous behavior is specified by the third parameter of the ajaxRequest call). The implementation below uses asynchronous AJAX. The drawback of using the synchronous version is that a network response problem could cause the browser to hang.

```javascript
// Make an AJAX request to the server to get who this incident would be assigned to given the current values in the record. This runs the assignment rules that have been defined in System Policy and returns the assigned_to and the assignment_group

function onChange(control, oldValue, newValue, isLoading){
    if(isLoading){return;
    // No change, do not do anything
    }

    // Construct the URL to ask the server for the assignment
    var url ="xmlhttp.do?sysparm_processor=AJAXAssignment&sys_target=incident";
    var uv = gel('sys_uniqueValue');
    if(uv){
        url +="&sys_uniqueValue="+ uv.value;
    // Make the AJAX request to the server and get the response
    var serial = g_form.serialize();
    // get all values currently assigned to the incident
    var response = ajaxRequest(url, serial,true, responseFunc);
    }

    // This callback function handles the AJAX response.
    function responseFunc(response){
        var item= response.responseXML.getElementsByTagName("item")[0];
        // Process the item returned by the server
        if(item){
            // Get the assigned_to ID and its display value and put them on the form
            var name=item.getAttribute("name");
            var name_label =item.getAttribute("name_label");
            if(name_label & name){
                g_form.setValue('assigned_to',name, name_label);
            }else{
                g_form.setValue('assigned_to','','');
            // Get the assignment_group ID and its display value and put them on the form
            var group =item.getAttribute("group");
            var group_label =item.getAttribute("group_label");
            if(group_label & group){
                g_form.setValue('assignment_group', group, group_label);
            }else{
                g_form.setValue('assignment_group','','');
            }}

    }
}
```

Custom approval UI macro

Type: UI macro

The following option illustrates how to obtain more detail from the My Approvals view of an Execution Plan by creating a new UI Macro.
- Navigate to **System UI** and click **UI Macros**.
- Rename the existing "approval_summarizer_sc_task" to something like "approval_summarizer_sc_task_old" and deactivate it.
- Create a new one using the same name ("approval_summarizer_sc_task"). The name should basically tell you what the macro does and to what it applies. In this case, we're replacing an existing one so we decided to re-use the existing name.

Then copy the xml script at the bottom of this article into the xml code window in the new UI Macro. This is great way to give some detail to an approver when you are doing line item approvals using approval tasks within the Service Catalog Execution Plans.

**Different ways**

**Old way**

This is the view you see in **My Approvals** when using an approval task using the old method.
Notice there is not much detail telling the approver what they are actually approving. You can see the short description of the task but not much information about the item.

**New way**

This is the view you will see if you use the xml script below in place of the OOB (out-of-box) UI Macro.

Using this method you can see details much like the request approval. You have a link into the item ordered, a short description (which contains the ability to expand the variables from the item), price, quantity and the total price. This helps the approver in that it shows more detail. They can now see what they are actually approving.

**Useful field scripts**

This is a searchable version of the useful field customization scripts.
**Caution:** The customization described here was developed for use in specific instances, and is not supported by ServiceNow Technical Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

For an easy-to-navigate version, visit the Useful Scripts portal.

**AKA Incident Template, Auto Assignments, Quick Calls, Call Script, Auto Populate**

Let’s say you want to auto-fill your Short Description based on the selected Subcategory. First, create a lookup table, then populate the key field, in this case the Subcategory and the autofilled field, Short Description. So let’s say your table had a record with Subcategory = Password and Short Description = Password Reset. When the user selects the Subcategory of Password on the Incident form, a client script looks up the matching record and sets Short Description equal to Password Reset.

Client script settings... **Type** = onChange, **Table name** = incident, **Field name** = Subcategory

```javascript
function onChange(control, oldValue, newValue, isLoading){
  if(isLoading){return;}
  var newrec = gel('sys_row');
  //Check if new record
  if (newrec.value == -1) {
    var lookup = new GlideRecord('u_short_desc_lookup');
    lookup.addQuery('u_subcategory', g_form.getValue('subcategory'));
    lookup.query();
    var temp; //temp var - reusable
    if(lookup.next()){
      temp = lookup.u_short_description;
      if(null != temp) {
        //Set the form value from lookup if there is a lookup value
        g_form.setValue('short_description', temp);
      } else {
        //If a lookup record does not exist based on lookup.addQuery
        //Then set to UNDEFINED or NULL depending on type
        g_form.setValue('short_description','');
      }
    }
  }
}
```

You could populate multiple fields or even pull Call Script questions into the Comments field so call center personnel gather good information to pass on to a technician. There are already Assignment Rules, Templates, and Wizards built in that perform similar functions.

**Disable HTML tags in descriptions**

Description: This code disables HTML tags in descriptions and short descriptions by substituting the tags with harmless versions that won’t execute.

```javascript
function doit(){
  var desc = current.description.toString();
  var shdesc = current.short_description.toString();
  if(desc.indexOf('script>')>-1|| shdesc.indexOf('script>')>-1){
    desc = desc.replace(/<script>/g,"(script)");
    current.description = desc.replace(/</script>/g,"(/script)"");
    shdesc = shdesc.replace(/<script>/g,"(script)"");
    current.short_description = shdesc.replace(/</script>/g,"(/script)"");
  }
}
```
Eliminate leading and trailing spaces in fields

Table: sys_user

Description: This example of the script trims trailing and leading spaces in the FirstName and LastName fields of sys_user.

doit();

function doit(){
    var gr = new GlideRecord('sys_user');
    gr.query();
    while(gr.next()){
        if((gr.first_name.toString().length!=
            gr.first_name.toString().trim().length) ||
            (gr.last_name.toString().length!=
            gr.last_name.toString().trim().length)){
            gr.first_name = gr.first_name.toString().trim();
            gr.last_name = gr.last_name.toString().trim();
            gr.autoSysFields(false);
            gr.update();}
    }
}

Make a field label flash

Type: Client script

Description: The following example is for the number field on incident. The label will flash for two seconds:

g_form.flash("incident.number","#FFFACD",0);

The arguments for the flash method are as follows:

1. tablename.fieldname
2. RGB color or acceptable CSS color like "blue" or "tomato"
3. integer that determines how long the label will flash:

   - Use 2 for a 1-second flash
   - Use 0 for a 2-second flash
   - Use -2 for a 3-second flash
   - Use -4 for a 4-second flash

Do not specify this argument if you want the field label simply colored the specified color.

Make field label bold

Type: Client script

Description: This script makes the label of a particular field (in this case, Short Description on the Incident Table) bold.

function onLoad(){
    var l = g_form.getLabel('incident.short_description');
    l.style.fontWeight='bold';
}

Make fields read-only

Type: Client script
Table: Incident

Description: This onLoad client script makes fields read-only. For this example, the script makes the following fields on the Incident table read-only: **Incident state**, **Impact**, **Urgency**, **Priority**, **Configuration item**, and **Assigned to**. It also removes the magnifying glass for the read-only Reference Fields (**Configuration item** and **Assigned to**).

```javascript
function onLoad(){
    var incidentState = g_form.getValue('incident_state');
    if(incidentState == '6' || incidentState == '7'){
        g_form.setReadonly('incident_state',true);
        g_form.setReadonly('impact',true);
        g_form.setReadonly('urgency',true);
        g_form.setReadonly('priority',true);
        g_form.setReadonly('cmdb_ci',true);
        g_form.setReadonly('assigned_to',true);}
}
```

**Set current date/time in field**

Type: Client script

Description: You can use the following two lines to set the current date and time in a date/time field. This bypasses the problem of getting the value into the proper format and proper timezone.

```javascript
var ajax = new GlideAjax('MyDateTimeAjax');
ajax.addParam('sysparm_name','nowDateTime');
ajax.getXML(function(){
    g_form.setValue('put your field name here', ajax.getAnswer());});
```

For more information on running server side scripts with the client, refer to **GlideAjax**.

**System script include**

```javascript
// Be sure the "Client callable" checkbox is checked
var MyDateTimeAjax = Class.create();
MyDateTimeAjax.prototype = Object.extendsObject(AbstractAjaxProcessor,{
    nowDateTime: function(){
        return gs.nowDateTime();}
});
```

**Toggle timer field by field name**

Type: Client script

Description: Toggles the timer field based on a particular field name.

```javascript
function toggleTimerByFieldName(fieldName){
   //Step 1: Find the timer object
   //timeObjectName: the timer objects name as it would normally be referenced
   //timeObjectHidden: the hidden input node in the field td
   //timeObjectParent: the parent td node containing the field and it's constituent nodes
   //timeObjectFields: anchor tag with onclick to stop timer

   var timeObjectName = fieldName;
   var timeObjectHidden = gel(timeObjectName);
   var timeObjectParent;
   var timeObjectFields;

   //Step 2: simulate click stop button
   var timeObjectParent;
   var timeObjectFields;
```
Modify GlideDateTime field value

Type: A server side script that accesses a GlideDateTime field.

Description: Given a GlideDateTime field or script object, show a variety of ways to easily modify value. The same concept also applies to the GlideDate object.

```javascript
// You first need a GlideDateTime object
// this can be from instantiating a new object "var gdt = new GlideDateTime()"
// or getting the object from a GlideDateTime field
// getting the field value (for example: var gdt = current.start_date)
// only returns the string value, not the object
// to get the object use var gdt = current.start_date.getGlideObject();
// now gdt is a GlideDateTime object
var gdt = current.start_date.getGlideObject();

// All methods can use negative values to subtract intervals

// add 1 hour (60 mins * 60 secs)
gdt.addSecondsLocalTime(3600);

// add 1 day
gdt.addDaysLocalTime(1);

// subtract 1 day
gdt.addDaysLocalTime(-1);

// add 3 weeks
gdt.addWeeksLocalTime(3);

// subtract 6 months.
gdt.addMonthsLocalTime(-6);

// add 1 year, representing the date and time using the UTC timezone instead of the local user's timezone.
gdt.addYearsUTC(1);
```
Useful scheduling scripts

A business rule script specifies the actions that the business rule takes. Scripts commonly include predefined global variables to reference items in your system, such as the current record. Global variables are available to all business rules.

Caution: The customization described here was developed for use in specific instances, and is not supported by ServiceNow Technical Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

Calculate duration given a schedule

Type: Before update/insert business rule

Table: Incident

Description: A Business Duration calculates the Open to Close duration on an incident based on the particular Schedules. If there is no schedule specified, the script will simply use the first schedule returned by the query.

Script example:

The example below sets the resolved duration when the incident state moves to resolved.

```javascript
if(current.incident_state==6){
  var dur = calcDurationSchedule(current.opened_at, current.sys_updated_on);
  current.u_resolved_duration= dur;
}

function calcDurationSchedule(start, end){
  // Get the user
  var usr = new GlideRecord('sys_user');
  usr.get(gs.getUserID());
  // Create schedule - pass in the sys_id of your standard work day schedule and pass in the users timezone
  var sched = new GlideSchedule('08fcd0830a0a0b2600079f56b1adb9ae',usr.time_zone);
  // Get duration based on schedule/timezone
  return(sched.duration(start.getGlideObject(), end.getGlideObject()));}
```

Check upcoming termination dates

Type: Scheduled script

Description: This script checks nightly for termination dates on contracts coming up in 90, 50, or 10 days (depending on the contract duration field).

Script example:

```javascript
function contractNoticeDue(){
  var gr = new GlideRecord("contract");
  gr.addQuery("u_contract_status","Active");
  gr.query();
  while(gr.next()){
    // Check uppercase for contract duration and contract status
    if((gr.u_termination_date<=
      gs.daysAgo(-90))&
      (gr.u_contract_duration=="Long"){
      gr.u_contract_status="In review";
      }elseif((gr.u_termination_date<=
        gs.daysAgo(-50))&
        (gr.u_contract_duration=="Medium"){
      gr.u_contract_status="In review";
      }elseif((gr.u_termination_date<=
        gs.daysAgo(-10))&
        (gr.u_contract_duration=="Short"){
      gr.u_contract_status="In review";
    }
  }
}
Use scripts in business rules to accomplish common tasks such as:

- Comparing two date fields.
- Parsing XML payloads.
- Aborting a database action in a business rule.

With scripts, you can also:

- Specify the operation that triggers the business rule.
- Use the scratchpad with display business rules to change form values just before a user loads the form.
- Use the OR condition like you would in a condition builder.

You can also utilize the system's scripting functionality available for server-side scripts.

You can use options on the Business Rules form to build conditions, set field values, and display alert messages without needing to write a script.

**Abort a database action in a business rule**

During a before business rule script, you can cancel or abort the current database action using the `current.setAbortAction(true)` method.

For example, if the before business rule is executed during an insert action, and you have a condition in the script that calls `current.setAbortAction(true)`, the new record stored in current is not created in the database.

**Add autofill functionality**

Add autofill functionality is also called incident template, auto assignments, quick calls, call script, or auto populate.

Let's say you want to auto-fill your **Short Description** based on the **Subcategory** selected. First, create a lookup table, then populate the key field, in this case **Subcategory** and the auto-filled field, **Short Description**. So let's say your table had a record with **Subcategory** = **Password** and **Short Description** = **Password Reset**. When the user selects the subcategory of **Password** on the Incident form a client script looks up the matching record and sets short description equal to **Password Reset**. Client script settings... **Type** = **onChange**, **Table name** = **incident**, **Field name** = **Subcategory**.

```javascript
function onChange(control, oldValue, newValue, isLoading) {
  if (isLoading) { return; }
  var newrec = gel('sys_row');
  //Check if new record
  if (newrec.value == -1) { 
    var lookup = new GlideRecord('u_short_desc_lookup');
    lookup.addQuery('u_subcategory', g_form.getValue('subcategory'));
    lookup.query();
    var temp; //temp var - reusable
    if (lookup.next()) { 
      temp = lookup.u_short_description;
      if (null != temp) { //Set the form value from lookup if there is a lookup value
        g_form.setValue('short_description', temp);
      } else {
        g_form.setValue('short_description', '');
      }
    } else {
      //If a lookup record does not exist based on lookup.addQuery
      //Then set to UNDEFINED or NULL depending on type
  ```
You could populate many fields or even pull in call script questions into the Comments field so call center personnel gather good information to pass on to a technician. There are already Assignment Rule, Templates and Wizards built in that perform similar functions.

Example script: A default before-query business rule
You can use a query business rule that executes before the database query is made to prevent users from accessing certain records.

**Caution:** The customization described here was developed for use in specific instances, and is not supported by ServiceNow Technical Support. This method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community forum.

Consider the following example from a default business rule that limits access to incident records.

**Default business rule limits access to incident records**

<table>
<thead>
<tr>
<th>Name</th>
<th>Table</th>
<th>When</th>
</tr>
</thead>
<tbody>
<tr>
<td>incident query</td>
<td>Incident</td>
<td>before, query</td>
</tr>
</tbody>
</table>

**Example script**

This example prevents users from accessing incident records unless they have the itil role are listed in the Caller or Opened by field. So, for example, when self-service users open a list of incidents, they can only see the incidents they submitted.

```javascript
if(!gs.hasRole("itil")&& gs.isInteractive()){  
    var u = gs.getUserID();  
    var qc = current.addQuery("caller_id", u).addOrCondition("opened_by", u).addOrCondition("watch_list", "CONTAINS", u);  
    gs.print("query restricted to user: "+ u);} 
```

**Note:** You can also use access controls to restrict the records that users can see.

**Schedule script for weekdays**

Type: Business Rules/Client Scripts

This script schedules the script for weekdays. Insert any script where it says ‘Your Script Here.’

```javascript
var go ='false';
var now =new Date();

// Correct time zone, which is by default GMT -7
now.setHours(now.getHours()+8);
var day = now.getDay();

// No go on Saturday or Sunday
if(day !=0&& day !=6)
    
// (your script here)
```
Set date field according to current date

This script sets a date field depending on the current day of the week. In this example, if the day is Monday through Wednesday, it sets the date to this coming Monday; otherwise it sets the date field to next Monday.

```javascript
function setCabDate()
{
    var today = new Date();
    var thisDay = today.getDay();
    // returns 0 for Sunday, 1 for Monday, etc. thru 6 for Saturday.
    var thisMon = new GlideDateTime();
    thisMon.setDisplayValue(gs.beginningOfThisWeek());
    var nextMon = thisMon.getNumericValue();
    nextMon += (1000*60*60*24*7);

    if((thisDay < 4) && (thisDay > 0))
    // if today is Mon thru Wed (thisDay = 1, 2, or 3), set cab to this coming Monday.
        current.u_req_cab_rev_date.setDateNumericValue(thisMon.getNumericValue());
    elseif((thisDay >= 4) || (thisDay == 0))
    // if today is Thurs thru Sun (thisDay = 4, 5, 6, or 0), set cab to next Monday.
        current.u_req_cab_rev_date.setDateNumericValue(nextMon);
}
```

To validate the input of all date/time fields, you can use the following in a validation script (System Definition > Validation Scripts). Because the date/time format is hard coded in this script, it must match your instance's date/time format. If your instance's date/time format changes, you must update your validation script.

Set the validation script's type to Date/Time. Then, with this validation script, if a user enters an incorrect format in a date/time field, they will receive an error message.

```javascript
function validate(value){
    // empty fields are still valid dates
    if(!value)return true;

    // We "should" have the global date format defined always defined. but there's always that edge case...
    if(typeof g_user_date_time_format !== 'undefined')
    return isDate(value, g_user_date_time_format);

    // if we don't have that defined, we can always try guessing
    return parseDate(value) !== null;}
```
Sample ASP Script for unencrypted single sign-on

This sample ASP .NET code creates a simple authentication portal and passes an unencrypted HTTP header as a URL parameter.

This sample assumes:

- The web server supports ASP .NET
- The target instance is `https://<instance name>.service-now.com/`
- SiteMinder or another single sign-on application has pre-authenticated the user
- The target instance expects an HTTP header of SM_USER

Change the ASP code to redirect users to the proper instance and create the proper HTTP header.

```html
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat = "server" >
<title >Portal Page Login </ title >

<%--    <meta http-equiv = "REFRESH" content = "0;url=https://<instance name>.service-now.com/">--%>
<script runat = "server" >

public void go_to(object sender, EventArgs e)
{
    ///Send URL parameters
    String URL = urlBox.Text + "?SM_USER=" + userNameBox.Text;
    Response.Redirect(URL);
}

</ script >

</ head >
<body >
<form id = "form1" runat = "server" >
<h2 >< b > Portal Page Login  </ b >< / h2 >

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
```
Validate date and time

To validate the input of all date/time fields, you can use the following in a validation script (System Definition > Validation Scripts).

Because the date/time format is hardcoded in this script, it must match your instance's date/time format. If your instance's date/time format changes, you must update your validation script.

Set the validation script's type to "glide_date_time". Then, with this validation script, if a user enters an incorrect format in a date/time field, they will receive an error message.

```javascript
function validate (value ) { if ( !value ) { return true ; } return (getDateFromFormat (value , 'yyyy-MM-dd HH:mm:ss' ) != 0 ) ;
}
```
Calculating duration

Often you may need to provide users with a way to specify when a task or process is due. Using the DurationCalculator script include, you can calculate the due date, using either a simple duration or relative duration.

For information on schedules, which you can use as inputs to DurationCalculator methods, see Schedules.

This script demonstrates how to use DurationCalculator to compute a due date.

```javascript
/**
 * Demonstrate the use of DurationCalculator to compute a due date.
 * 
 * You must have a start date and a duration. Then you can compute a
 * due date using the constraints of a schedule.
 */
gs.include('DurationCalculator');
executeSample();

/**
 * Function to house the sample script.
 */
function executeSample()
{
  // First we need a DurationCalculator object.
  var dc = new DurationCalculator();

  // --------------- No schedule examples ------------------

  // Simple computation of a due date without using a schedule. Seconds/
  // are added to the start date continuously to get to a due date.
```
dc.setStartDateDateTime("5/1/2012");if(!dc.calcDuration(2*24*3600)){ // 2 days
gs.log("*** Error calculating duration");return;
gs.log("calcDuration no schedule: "+ dc.getEndDateTime()); // "2012-05-03 00:00:00" two days later

// Start in the middle of the night (2:00 am) and compute a due date 1 hour in the future. Without a schedule this yields 3:00 am.
dc.setStartDateDateTime("5/3/2012 02:00:00");if(!dc.calcDuration(3600)) {
gs.log("*** Error calculating duration");return;
gs.log("Middle of night + 1 hour (no schedule): "+ dc.getEndDateTime()); // No scheduled start date, just add 1 hour
}

// -------------- Add a schedule to the date calculator
_____
addSchedule(dc);

// Start in the middle of the night and compute a due date 1 hour in the future. Since we start at 2:00 am the computation adds the 1 hour from the start of the day, 8:00am to get to 9:00am
dc.setStartDateDateTime("5/3/2012 02:00:00");if(!dc.calcDuration(3600)){
gs.log("*** Error calculating duration");return;
gs.log("Middle of night + 1 hour (with 8-5 schedule): "+ dc.getEndDateTime()); // 9:00 am
}

// Start in the afternoon and add hours beyond quitting time. Our schedule says the work day ends at 5:00pm, if the duration extends beyond that, we roll over to the next work day. In this example we are adding 4 hours to 3:00pm which gives us 10:00 am the next day.
dc.setStartDateDateTime("5/3/2012 15:00:00");if(!dc.calcDuration(4*3600)) {
//
    gs.log("*** Error calculating duration");return;
    gs.log("Afternoon + 4 hour (with 8-5 schedule): "+ dc.getEndDateTime()); // 10:00 am.
}

// This is a demo of adding 2 hours repeatedly and examine the result. This is a good way to visualize the result of a due date calculation.
dc.setStartDateDateTime("5/3/2012 15:00:00");for(var i=2; i<24; i+=1) {
if(!dc.calcDuration(i*3600)) {
    gs.log("*** Error calculating duration");return;
    gs.log("add "+ i +" hours gives due date: "+ dc.getEndDateTime());
}

// Setting the timezone causes the schedule to be interpreted in the specified timezone. Run the same code as above with different timezone. Note that the 8 to 5 workday is offset by the two hours as specified in our timezone.
dc.setTimeZone("GMT-2");
dc.setStartDateDateTime("5/3/2012 15:00:00");for(var i=2; i<24; i+=1) {
if(!dc.calcDuration(i*3600)) {
    gs.log("*** Error calculating duration");return;
    gs.log("add "+ i +" hours gives due date (GMT-2): "+ dc.getEndDateTime());
}

/**
 * Add a specific schedule to the DurationCalculator object.
 *
 * @param durationCalculator An instance of DurationCalculator
 */
function addSchedule(durationCalculator) {
    // Load the "8-5 weekdays excluding holidays" schedule into our duration calculator.
    var scheduleName = "8-5 weekdays excluding holidays";
    var grSched = new GlideRecord('cmn_schedule');
    grSched.addQuery('name', scheduleName);
    grSched.query();if(!grSched.next()){

Simple duration vs relative duration

How much work is required to complete a task can be expressed as a "relative duration". Relative duration determines the expected due date and time relative to the starting time. Examples of relative durations include "Next business day by 4pm," or '2 business days by 10:30am.'

To calculate a relative duration, the calendar and time zone must be considered to determine what 'next business day' means since it is the calendar that defines which days are valid work days and the time zone will affect the result as well. As an example, consider 'Next business day by 4pm':
- If it is Monday at 12pm: Next business day by 4pm => Tuesday at 4pm
- If it is Friday at 2pm: Next business day by 4pm => the following Monday at 4pm

Note: Next business day is often defined by a starting day and time. For example, "next business day at 4pm if before 2pm" indicates that if the current time is after 2pm on a business day, then "Next business day" really means 2 business days since today does not count.

Calculating a simple duration

This business rule and script example demonstrate how to calculate a simple duration.

```javascript
var dur = new DurationCalculator();
dur.setSchedule(current.schedule);
dur.setStartDateTime("");
if(current.duration_type==""){
    dur.calcDuration(current.duration.getGlideObject().getNumericValue()/1000);}
else{
dur.calcRelativeDuration(current.duration_type);
}
current.end_date_time= dur.getDateTime();
current.work_seconds= dur.getSeconds();
```

This script demonstrates how to use DurationCalculator to calculate a simple duration.

```javascript
/**
 * Sample script demonstrating use of DurationCalculator to compute simple durations
 */
gs.include('DurationCalculator');
executeSample();

/**
 * Function to house the sample script.
 */
function executeSample(){
    // First we need a DurationCalculator object.
    var dc = new DurationCalculator();

    // Compute a simple duration without any schedule. The arguments
    // can also be of type GlideDateTime, such as fields from a GlideRecord.
    var dur = dc.calcScheduleDuration("5/1/2012","5/2/2012");
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
gs.log("calcScheduleDuration no schedule: "+ dur);
// 86400 seconds (24 hours)

// The above sample is useful in limited cases. We almost always want
to
// use some schedule in a duration computation, let's load a schedule.
addSchedule(dc);

// Compute a duration using the schedule. The schedule
// specifies a nine hour work day. The output of this is 32400 seconds,
or
// a nine hour span.
dur = dc.calcScheduleDuration("5/23/2012 12:00","5/24/2012 12:00");
gs.log("calcScheduleDuration with schedule: "+ dur);
// 32400 seconds (9 hours)

// Compute a duration that spans a weekend and holiday. Even though this
// spans three days, it only spans 9 work hours based on the schedule.
dur = dc.calcScheduleDuration("5/25/2012 12:00","5/29/2012 12:00");
gs.log("calcScheduleDuration with schedule spanning holiday: "+ dur);
// 32400 seconds (9 hours)

// Use the current date time in a calculation. The output of this is
// dependent on when you run it.
var now =new Date();
dur = dc.calcScheduleDuration("5/15/2012",new GlideDateTime());
gs.log("calcScheduleDuration with schedule to now: "+ dur);
// Different on every run.)

/**
 * Add a specific schedule to the DurationCalculator object.
 * 
 * @param durationCalculator An instance of DurationCalculator
 */
function addSchedule(durationCalculator){

    // Load the "8-5 weekdays excluding holidays" schedule into our duration
    // calculator.
    var scheduleName ="8-5 weekdays excluding holidays";
    var grSched =new GlideRecord('cmn_schedule');
    grSched.addQuery('name', scheduleName);
    grSched.query();if(!grSched.next()){gs.log('*** Could not find schedule "'+ scheduleName +""');
        return;}
    durationCalculator.setSchedule(grSched.getUniqueValue());}

**Calculating a relative duration**

An example of a relative duration calculation script.

This script calculates the relative duration for "Next day at 4pm if after 10am":

// Next day at 4pm if before 10am
var days =1;
if(calculator.isAfter(calculator.startDateTime,"10:00:00"))
    days++;
calculator.calcRelativeDueDate(calculator.startDateTime, days,"16:00:00");

This script demonstrates how to use DurationCalculator to calculate a relative duration.

/**
 * Sample use of relative duration calculation.
 * 
 */
gs.include('DurationCalculator');
executeSample();

/**
 * Function to house the sample script.
 */
function executeSample(){

    // First we need a DurationCalculator object. We will also use
    // the out-of-box relative duration "2 bus days by 4pm"
    var dc = new DurationCalculator();
    var relDur = "3bf802c20a0a0b52008e2859cd8abcf2";
    // 2 bus days by 4pm if before 10am
    addSchedule(dc);

    // Since our start date is before 10:00am our result is two days from
    // now at 4:00pm.
    dc.setStartDateTime("5/1/2012 09:00:00");
    if(!dc.calcRelativeDuration(relDur)){
        gs.log("*** calcRelativeDuration failed");
        return;
    }
    gs.log("Two days later 4:00pm: "+ dc.getEndTime());

    // Since our start date is after 10:00am our result is three days from
    // now at 4:00pm.
    dc.setStartDateTime("5/1/2012 11:00:00");
    if(!dc.calcRelativeDuration(relDur)){
        gs.log("*** calcRelativeDuration failed");
        return;
    }
    gs.log("Three days later 4:00pm: "+ dc.getEndTime());

    /**
     * Add a specific schedule to the DurationCalculator object.
     *
     * @param durationCalculator An instance of DurationCalculator
     */
    function addSchedule(durationCalculator){
        // Load the "8-5 weekdays excluding holidays" schedule into our duration
        // calculator.
        var scheduleName = "8-5 weekdays excluding holidays"
        var grSched = new GlideRecord('cmn_schedule');
        grSched.addQuery('name', scheduleName);
        grSched.query();
        if(!grSched.next()){ 
            gs.log("*** Could not find schedule "+ scheduleName +""");
            return;
        }
        durationCalculator.setSchedule(grSched.getUniqueValue(), "GMT");
    }

Elapsed time vs work time

Typically, setting a due date requires that you calculate the actual work time required for
completion rather than the total time that elapses until the due date.

SLAs, Workflow tasks, and approvals are examples of situations that require the actual work rather
than the total time that has elapsed. In this case, only the part of the day when work is performed
is considered when determining when the work is to be complete. For example, if a task is due in
10 hours, but the actual time to perform the task is restricted to a business day schedule, the 10
hours of work can only be done at most 8 hours each day. If this work starts at 10am on Monday, it
is expected to complete, or be due, on Tuesday at 12pm:

10am-5pm on Monday (6 hours) + 8am-12pm on Tuesday (4 hours)
How to implement a relative duration

You can implement a relative duration by creating the cmn_relative_duration table and the DurationCalculator script include.

Role required: admin

1. Create the cmn_relative_duration table.
2. Create the DurationCalculator script include.
3. Create a sample relative duration entry (for example, "Next business day by 4pm").
4. Add the needed fields to SLA tables to support relative durations.
5. Modify duration calculation for SLAs.
6. Modify SLA Percentage timer calculation for SLAs (this must use work_seconds).
7. Add schedule fields to the Workflow: Schedule and Timezone (selected based on the field from workflow table).
8. Add duration support fields to the Workflow Task activity.
9. Implement duration calculation script for the task activity.

The relative duration table and the DurationCalculator methods

The cmn_relative_duration table supports the definition of a due date as either a duration of time or a relative duration.

This table consists of two fields: "name" and "script." The "script" field contains the relative duration calculation script. This script includes the "calculator" variable, which is used to calculate the due date.

The DurationCalculator script include can be used to perform the duration calculations. The following are methods that are available in this script include.

**DurationCalculator script include table**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setSchedule(String schedID, (String timezone))</td>
<td>Sets the schedule and time zone to be used for calculating the due date.</td>
</tr>
<tr>
<td>setStartDateTime(GlideDateTime start)</td>
<td>Sets the start time for the duration calculations. If 'start' is blank, uses current date/time.</td>
</tr>
<tr>
<td>calcDuration(int seconds)</td>
<td>Calculates the end date and time. Upon completion the this.endDateTime and this.seconds properties will be set to indicate the results of the calculation.</td>
</tr>
<tr>
<td>calcRelativeDuration(String relativeDurationID)</td>
<td>Calculates the duration using the specified relative duration script. Upon completion the this.endDateTime and this.seconds properties will be set to indicate the results of the calculation.</td>
</tr>
<tr>
<td>getEndDateTime()</td>
<td>Gets the this.endDateTime property that was set by calcDuration/calcRelativeDuration indicating the end date and time for the duration.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getSeconds()</td>
<td>Gets the this.seconds property that was set by calcDuration/calcRelativeDuration indicating the total number of seconds of work to be performed for the duration.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This is the total work time, not the total time between start and end times and may be used to determine percentages of the work time.</td>
</tr>
<tr>
<td>getTotalSeconds()</td>
<td>Gets the this.totalSeconds property that was set by calcDuration/calcRelativeDuration indicating the total number of seconds between the start and end times of the duration.</td>
</tr>
</tbody>
</table>

The following functions are used in relative duration scripts:

**Relative duration script functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean isAfter(GlideDateTime dt, String time)</td>
<td>Is ‘time’ of day after the time of day specified by ‘dt’? dt, if blank, uses current date/time. time is in ‘hh:mm:ss’ in 24-hour format.</td>
</tr>
<tr>
<td>calcRelativeDueDate(GlideDateTime start, int days, String endTime)</td>
<td>Calculates the due date starting at ‘start’ and adding ‘days’ using the schedule and time zone. When we find the day that the work is due on, set the time to ‘endTime’ of that day. Upon completion, this.endDateTime and this.seconds properties will be set to indicate the results of the calculation. If endTime is blank, use end of the ending work day.</td>
</tr>
</tbody>
</table>

**JavaScript engine upgrade**

The JavaScript engine used to evaluate scripts has been upgraded to support the ECMAScript5 standard.

There are no plugins or properties needed to install the new JavaScript engine, and all Helsinki instances use the new JavaScript engine. The new JavaScript engine is part of the platform; and Express, MID Server, and Edge Encryption use the new JavaScript engine.

The benefits include the following.

- You can use modern library code, such as lodash.js and moment.js.
- Your script will follow standard ECMAScript5 behavior.

The new JavaScript engine is based upon Rhino version 1.7 R5.

**What you need to know**

The new JavaScript engine provides an improved environment for developing scripts.

- Legacy code continues to work.
- Compatibility mode supports the legacy modifications to the old JavaScript engine.
ServiceNow    Kingston    Now Platform Custom Business Applications

- All scripts created prior to the Helsinki release and global scripts run in compatibility mode.
- New applications created in Helsinki default to run in ES5 standards mode.

**JavaScript modes**

To support existing scripts and new scripts developed to the ECMAScript5 standard, the JavaScript engine has two modes. The modes are Compatibility Mode and ES5 Standards Mode. The JavaScript engine dynamically determines which mode to use on a script-by-script basis.

**Compatibility mode**

Compatibility mode is used for all scripts developed prior to the Helsinki release, and all global scripts. Compatibility mode has some differences from the old JavaScript engine.

**JSON support changes.**

- `JSON.stringify()` and `JSON.parse()` are now implemented using the ES5 Native JSON object.
- The new `JSON().encode()` and new `JSON().decode()` are still supported, but should only be used when the legacy behavior is required.

The use of 3rd party JavaScript libraries is not supported in Compatibility mode.

**ES5 Standards mode**

ES5 Standards mode is the default when you create new scoped scripts. This mode does not preserve the legacy behaviors in the pre-Helsinki JavaScript engine.

ES5 standards mode supports ECMAScript5 syntax and features, including the following.

- The `use strict` declaration
- Control over extensibility of objects
- Get and set properties on objects (accessors)
- Control over write-ability, configurability, and enumer-ability of object properties
- New Array and Date methods
- Native JSON support
- Support for modern third-party libraries such as lodash.js and moment.js

**Porting code to ES5 standards mode scripts**

ES5 standards mode catches errors that compatibility mode allows.

Things to watch for when porting code from existing scripts to new scoped scripts using ES5 standards mode.

ECMAScript5 evaluates the term `new Boolean(false)` to true. In compatibility mode, it evaluated to false.

ECMAScript5 throws an EcmaError when a non-existent property is referenced. In compatibility mode no error was thrown.

ECMAScript5 throws an EcmaError when a non-existent function is called. In compatibility mode, no error was thrown.
ECMAScript5 correctly handles new lines. In the past, a newline character after a comment was recognized, which is wrong. In this example, in compatibility mode, all three functions are called. In ECMAScript5, only the first function is called.

```javascript
var expr = doFoo();  // do foo
doBar();  // do bar
finish();   // all done
eval(expr);
```

ECMAScript5 correctly handles postfix increment and decrement. In this example, in compatibility mode, the variable `x` gets the incremented value, which is wrong.

```javascript
var x = gr.limit++;
```

### Script debugger

The Script Debugger allows application developers to debug server-side JavaScript.

The Script Debugger allows each application developer to:

- Have a dedicated debug transaction, which only applies to the current session.
- Set and remove breakpoints.
- Pause the current session at a breakpoint.
- Step through code line-by-line.
- Step into and out of function and method calls.
- View the value of local and global variables.
- View the value of private variables from function closures.
- View the call stack.
- View the transaction the system is processing.
- Turn off the script debugger to resume running paused scripts.

The Script Debugger can pause any server-side script that runs in an interactive transaction such as business rules, script includes, script actions, or UI actions that require a response in order to proceed. If the GlideSystem method `isInteractive()` would return true when running the script in context, then the Script Debugger can pause it.

**Note:** Some script objects such as script includes can be called from multiple contexts. When a business rule runs a script include on form submit that is an interactive transaction waiting on the form data to change before continuing. When a scheduled job runs the same script include that is a non-interactive background transaction that can also run other scripts simultaneously.

To debug client-side script, you can use browser-based developers tools.

Only users with the script_debugger role can access the Script Debugger.

A debugger transaction remains open as long as the user session is valid. If a user session logs out or times out, the system closes the debugger transaction.

### Parts of the Script Debugger interface

The Script Debugger interface displays information about breakpoints set, the call stack and line number of the currently executing script line, and details about variables and transactions.
Script Debugger Not Paused
### Parts of the Script Debugger

<table>
<thead>
<tr>
<th>User interface element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakpoints</td>
<td>Displays a list of the breakpoints set by script type, script name, and line number. The debugger updates this list as you add and remove breakpoints.</td>
</tr>
<tr>
<td>Call stack</td>
<td>Displays a list of script calls that preceded or invoked the current line number. This information is only visible when the debugger pauses on a breakpoint.</td>
</tr>
<tr>
<td>Transaction details</td>
<td>Displays information about the current transaction. This information is only visible when the debugger pauses on a breakpoint.</td>
</tr>
<tr>
<td>Status</td>
<td>Displays if the debugger is waiting for a breakpoint, paused on a breakpoint, or has encountered an exception.</td>
</tr>
<tr>
<td>User</td>
<td>Displays the name of the user who is running the current debugger session.</td>
</tr>
<tr>
<td>Coding pane header</td>
<td>Displays the script type and name of the script in the coding pane.</td>
</tr>
<tr>
<td>User interface element</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Breakpoint icon</td>
<td>Indicates the line number where the debugger pauses when evaluating the current script.</td>
</tr>
<tr>
<td>Pause debugging button</td>
<td>Stops any current debugging session, and disables the Script Debugger for the current user. The Script Debugger does not pause on breakpoints for the current user until it is restarted.</td>
</tr>
<tr>
<td>Resume script execution button</td>
<td>Advances from the current breakpoint to the next breakpoint. If there are no other breakpoints, the script runs to completion.</td>
</tr>
<tr>
<td>Step over next function call button</td>
<td>Advances past the method that is about to be called, executing the method as a single step.</td>
</tr>
<tr>
<td>Step into next function call</td>
<td>Advances to the first line of executed code within a method call. Stepping into a method updates the current position within the call stack. If the user does not have read access to the method call, then this control acts like step over instead.</td>
</tr>
<tr>
<td>Step out of current function</td>
<td>Exits from current method call and returns to the calling script from the call stack. If the user is not within a method call, then this control acts like step over instead.</td>
</tr>
<tr>
<td>Local</td>
<td>Displays a list of local scope JavaScript variable names and their values. This information is only visible when the debugger pauses on a breakpoint.</td>
</tr>
<tr>
<td>Closures</td>
<td>Displays a list of global scope JavaScript variable names and their values set by function closure. This information is only visible when the debugger pauses on a breakpoint.</td>
</tr>
<tr>
<td>Global</td>
<td>Displays a list of global scope JavaScript variable names and their values. This information is only visible when the debugger pauses on a breakpoint.</td>
</tr>
</tbody>
</table>

**Script Debugger step-through controls**

After the Script Debugger pauses a script, use the step-through controls to move between script lines and move between scripts in the call stack.

**Step-through controls**

<table>
<thead>
<tr>
<th>Control</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pause debugging - F2</td>
<td><img src="pause_icon.png" alt="Pause Icon" /></td>
<td>Stops any current debugging session, and disables the Script Debugger for the current user. The Script Debugger does not pause on breakpoints for the current user until it is restarted.</td>
</tr>
<tr>
<td>Start debugger - F2</td>
<td><img src="start_icon.png" alt="Start Icon" /></td>
<td>Enables the Script Debugger for the current user. The Script Debugger pauses on breakpoints.</td>
</tr>
<tr>
<td>Control</td>
<td>Icon</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Resume script execution - F9</td>
<td>![play]</td>
<td>Advances from the current breakpoint to the next breakpoint. If there are no other breakpoints, the script runs to completion.</td>
</tr>
<tr>
<td>Step over next function call - F8</td>
<td>![reload]</td>
<td>Advances to the next evaluated line of script based on current conditions. The Script Debugger skips any lines of code that do not need to run because their conditions are not met. For example, when the condition of an <strong>if</strong> statement is not true, the debugger skips the code block for the condition.</td>
</tr>
<tr>
<td>Step into next function call - F7</td>
<td>![arrow_down]</td>
<td>When the Script Debugger pauses on a method call, this control allows the user to advance to the first line of executed code within the method call. Stepping into a method updates the current position within the call stack. If the user does not have read access to the method call, then this control acts like step over instead.</td>
</tr>
<tr>
<td>Step out of current function - SHIFT+F8</td>
<td>![arrow_up]</td>
<td>When the Script Debugger pauses within a method call, this control allows the user to exit the current method call and return to the calling script from the call stack. If the user is not within a method call, then this control acts like step over instead.</td>
</tr>
</tbody>
</table>

**Launch the Script Debugger**

Developers can launch the Script Debugger from the application navigator, Studio, or from the syntax editor.

Role required:
- admin
- script_debugger

Select a path based on your starting point:

<table>
<thead>
<tr>
<th>Starting point</th>
<th>Navigation path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application navigator</td>
<td>Navigate to System Diagnostics &gt; Script Debugger.</td>
</tr>
<tr>
<td>Studio</td>
<td>Navigate to File &gt; Script Debugger.</td>
</tr>
<tr>
<td>Syntax Editor</td>
<td>Click the <strong>Script Debugger</strong> icon.</td>
</tr>
</tbody>
</table>

The system opens the Script Debugger in a new window.
Set or remove breakpoints

Set breakpoints to pause scripts at specific lines, and remove breakpoints when you are done debugging them.

Role required:
- admin
- script_debugger

Breakpoints belong to the developer who sets them. Developers must set and remove their own breakpoints.

1. Navigate to the server script to debug. For example, navigate to System Definition > Business Rules.

2. From the Syntax Editor, click the gutter next to a script line.

<table>
<thead>
<tr>
<th>Current state</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blank line</td>
<td>Click a blank line to set a breakpoint. The Script Debugger pauses the script at this line.</td>
</tr>
<tr>
<td>Breakpoint</td>
<td>Click a breakpoint to remove it.</td>
</tr>
</tbody>
</table>

3. From the Syntax Editor toolbar, click the Open Script Debugger icon. The system opens a Script Debugger window.

4. Trigger the script. For example, create a record to trigger an insert business rule script. The Script Debugger pauses the script on the first line containing a breakpoint, and the system displays a confirmation window.

5. Click Start Debugging. The system switches focus to the Script Debugger window and displays the target script paused at the first breakpoint.

6. When debugging is complete, remove breakpoints from the script.

Script Debugger status

The Script Debugger status determines what debugging actions are available and what information it can display.

The Script Debugger displays its status at the bottom left of the user interface.
### Status: EXECUTION_PAUSED

Sample Script Debugger status

#### Possible Script Debugger status values

<table>
<thead>
<tr>
<th>Status</th>
<th>Occurs when</th>
<th>Description</th>
<th>Actions available</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAITING FOR FIRST BREAKPOINT</td>
<td>The user opens a Script Debugger window or tab.</td>
<td>The Script Debugger is ready to pause script and display debugging information.</td>
<td>• Pause script at the first breakpoint in the call stack.</td>
</tr>
<tr>
<td>EXECUTION_PAUSED</td>
<td>• The Script Debugger pauses on a breakpoint.</td>
<td>The Script Debugger has paused on a line of code, and the user can debug the script.</td>
<td>• Resume processing until the Script Debugger reaches the next breakpoint. • Step through a script. • Display the call stack. • Display transaction information. • Display variable values.</td>
</tr>
<tr>
<td>WAITING FOR BREAKPOINT</td>
<td>The user resumes processing until the Script Debugger reaches the next breakpoint.</td>
<td>The Script Debugger is searching for the next line of code at which to pause. Users will typically never see this status because the Script Debugger changes status after it locates the next breakpoint or script line to evaluate.</td>
<td>• Pause script at the next breakpoint. • Pause script at the next script line requiring evaluation.</td>
</tr>
<tr>
<td>OFF</td>
<td>• The user pauses the Script Debugger.</td>
<td>The Script Debugger is inactive and does not pause scripts or display debugging information.</td>
<td>• Start the Script Debugger. • Open a Script Debugger window or tab.</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Log entries

Every time a debug transaction finishes executing, the system creates a log entry for it with a DEBUGGED prefix. For example:

```
2016-08-15 15:57:32 (197) Default-thread-3 900F510167112200C4098C7942415A75
*** End
#39, path: /my-app.do, user: admin, DEBUGGED total transaction time: 0:00:11.010,
transaction processing time: 0:00:11.010, network: 0:00:00.000, chars: 6,058, uncompressed
chars: 20,731, SQL time: 50 (count: 34), business rule: 0 (count: 0), phase 1
form length 56,464, largest chunk written: 10,428, request parms size: 40,
```

largest input read: 0

Transaction details

The Script Debugger displays transaction details for the current paused user session.

Transaction details are available in a dedicated resizeable section underneath the Call Stack on the bottom left of the Script Debugger.
The Script Debugger only displays transaction details when it pauses on a script. Developers can use transaction details to:

- Inspect the URL of the currently paused transaction.
- Inspect the request parameters for the currently paused transaction.
- Inspect network information about the current transaction.
- Inspect the user and session ID that initiated the debug transaction.

Available transaction details

The Script Debugger provides a standard set of transaction details for developers to debug and troubleshoot scripts.

Available transaction details

<table>
<thead>
<tr>
<th>Transaction element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>The URL of the currently paused transaction.</td>
</tr>
<tr>
<td>Request parameters</td>
<td>The list of request parameters for this transaction. Each transaction has its own list of request parameters, but record transactions typically include the field values used to insert, update, or delete a record.</td>
</tr>
<tr>
<td>instance</td>
<td>The instance name.</td>
</tr>
<tr>
<td>address</td>
<td>The IP address of the end-user client system.</td>
</tr>
<tr>
<td>session</td>
<td>The user session ID.</td>
</tr>
<tr>
<td>forward</td>
<td>The IP address of the load balancer.</td>
</tr>
<tr>
<td>query count</td>
<td>The number of database queries the Script Debugger has made.</td>
</tr>
<tr>
<td>thread</td>
<td>The name of the thread running the Script Debugger instance.</td>
</tr>
<tr>
<td>transactionid</td>
<td>The Sys ID of the current transaction.</td>
</tr>
<tr>
<td>token</td>
<td>The Script Debugger token of the currently paused transaction. The system uses this token to identify different Script Debugger instances.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the currently paused transaction. You can use this name to identify transactions in the logs.</td>
</tr>
<tr>
<td>processor</td>
<td>The name of the processor processing the current transaction, if present.</td>
</tr>
<tr>
<td>method</td>
<td>The HTTP request method the currently paused transaction uses.</td>
</tr>
<tr>
<td>startTime</td>
<td>The date-time stamp when the Script Debugger instance started.</td>
</tr>
<tr>
<td>page</td>
<td>The current table or UI page associated with the transaction.</td>
</tr>
<tr>
<td>user</td>
<td>The user who triggered the debug transaction.</td>
</tr>
<tr>
<td>nodeid</td>
<td>The Sys ID of the node running the Script Debugger instance.</td>
</tr>
</tbody>
</table>
Script Debugger multiple developer support

The Script Debugger allows multiple developers to debug their own transactions without affecting each other.

The Script Debugger only allows developers to see and interact with items related to their current debugging session such as:

- Breakpoints
- Call stack
- Transactions
- Status

The Script Debugger prevents one developer from seeing or modifying another debug session. Administrators, however, can impersonate another user, open the Script Debugger, and debug transactions generated by the impersonated user.

The Script Debugger displays the debug session user at the bottom left of the user interface.

User: System Administrator

Sample Script Debugger user

Concurrent Script Debugger usage

By default, the system supports debugging \([(\text{The number of semaphores on the \text{instance}}) / 4]\) concurrent transactions. Administrators can specify the number of concurrent transactions the system can debug by setting the `glide.debugger.config.max_node_concurrency` system property. The system can debug up to \([(\text{The number of semaphores on the \text{instance}}) - 2]\) concurrent transactions.

Administration of debugging sessions

Debugging sessions can remain actively debugging (in the EXECUTION_PAUSED or WAITING_FOR_BREAKPOINT statuses) until:

- The user pauses the Script Debugger.
- The user closes the Script Debugger.
- The user session ends.

Administrators can view the currently running debugger sessions by navigating to the page `xmlstats.do`.

Administrators can stop all currently running debugging sessions by navigating to the page `debugger_reset.do`. Only users with the admin role can access this page.

Script Debugger impersonation support

You can use the Script Debugger while impersonating another user, but only if the impersonated user has the script_debugger role and has read access to the target script.

While impersonating another user, you can:

- See and change breakpoints that belong to the impersonated user.
- View and pause on scripts that the impersonated user has read access to.

The Script Debugger step-through controls also use the read access of the impersonated user. For example, if the impersonated user does not have read access to a function in the call stack, any **Step into** action instead becomes a **Step over** action.

The impersonated debugging session lasts until:

- You stop impersonating the user.
- You log out or the user session ends.
- You pause the Script Debugger.
- You close the Script Debugger.

**Script Debugger Scripts - Background support**

The Scripts - Background module does not support setting breakpoints directly in the script field. You can however, set breakpoints in the script objects called or triggered by the Scripts - Background module.

While running arbitrary JavaScript code in the **Scripts - Background** module, the Script Debugger can only pause scripts when you:

- Call a script include containing breakpoints.
- Trigger a business rule containing breakpoints.
- Trigger a script action containing breakpoints.

**Domain separation and Script Debugger**

This is an overview of domain separation and the Script Debugger feature. Domain separation allows you to separate data, processes, and administrative tasks into logical groupings called domains. You can then control several aspects of this separation, including which users can see and access data.

**Overview**

**Support: Level 1**

Domain separation is supported in this application. Not all ServiceNow applications support domain separation; some include limitations on the data and administrative settings that can be domain separated. To learn more, see Application support for domain separation.

**How domain separation works in Script Debugger**

Script Debugger is not a full application but rather, a feature in the Platform suite, meaning it works alongside other features, including domain separation.

**Session debug**

Enable session debugging to display debugging messages in the user interface.

You can enable all areas for abundant logging on the bottom of each page load, or you can enable each module one by one, for more specific information about what is happening during this session, and specifically, for the previous transaction. Select session debug options under **System Diagnostics > Session Debug**. When enabled, session debugging is active during the user session or until disabled.

The system provides the following session debugging options.
## Session debug options

<table>
<thead>
<tr>
<th>Debug option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable All</td>
<td>Displays all available debugging messages.</td>
</tr>
<tr>
<td>Disabled All</td>
<td>Stops displaying all debugging messages.</td>
</tr>
<tr>
<td>Debug Business Rule</td>
<td>Display debugging messages for business rules. If there are business rules from multiple applications affecting a table or record, the system displays which application the business rule comes from.</td>
</tr>
<tr>
<td>Debug Business Rule (Details)</td>
<td>Displays debugging messages for business rules and any changes made by business rules. If there are business rules from multiple applications affecting a table or record, the system displays which application the business rule comes from.</td>
</tr>
<tr>
<td>Debug Log</td>
<td>Displays all log entries.</td>
</tr>
<tr>
<td>Debug Date/Time</td>
<td>Displays Date/Time failures when inputs do not match required formats.</td>
</tr>
<tr>
<td>Debug SQL</td>
<td>Displays debugging messages for SQL queries.</td>
</tr>
<tr>
<td>Debug SQL (Detailed)</td>
<td>Displays debugging messages for SQL statements and any changes made by SQL statements.</td>
</tr>
<tr>
<td>Debug Security</td>
<td>Displays debugging messages for access controls. If there are access controls from multiple applications affecting a table or record, the system displays which application the access controls comes from.</td>
</tr>
<tr>
<td>Debug Escalations</td>
<td>Displays debugging messages for SLA and SLO escalations.</td>
</tr>
<tr>
<td>Debug Metric Statistics</td>
<td>Displays an aggregate view of performance data (slow transactions, scripts, queries, events, and mutexes). These aggregate metrics are sorted by transaction, to help identify items that affect page performance.</td>
</tr>
<tr>
<td>Debug Text Search</td>
<td>Displays debugging messages for search result relevance and indexing.</td>
</tr>
<tr>
<td>Debug UI Policies</td>
<td>Displays debugging messages for UI policies.</td>
</tr>
<tr>
<td>Disable UI Policies Debug</td>
<td>Stops displaying debugging messages for UI policies.</td>
</tr>
<tr>
<td>Debug Data Policies</td>
<td>Displays debugging messages for data policies.</td>
</tr>
<tr>
<td>Debug Quotas</td>
<td>Displays debugging messages for transaction quotas.</td>
</tr>
<tr>
<td>Debug Homepage Render</td>
<td>Displays debugging messages for homepages.</td>
</tr>
<tr>
<td>Debug Scopes</td>
<td>Displays debugging messages for entering or exiting application scopes when running script objects.</td>
</tr>
</tbody>
</table>
Debugging applications

Application developers can display debug messages about configuration records to help them troubleshoot issues. The Debug Scopes module provides information about the system switching between custom applications to run server-side scripts.

The system offers the following debugging options to help application developers determine how applications affect configuration records.

### Application debug options

<table>
<thead>
<tr>
<th>Debugging option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug Business Rule</td>
<td>Use this module to determine which application's business rules are running against tables. The system only displays application information if business rules from different application scopes run on the same table.</td>
</tr>
<tr>
<td>Debug Business Rule (Details)</td>
<td>Use this module to determine the results of running business rules against tables. The system only displays application information if business rules from different application scopes run on the same table.</td>
</tr>
<tr>
<td>Debug Security</td>
<td>Use this module to determine which application's access controls apply to a given table or record.</td>
</tr>
<tr>
<td>Debug Scopes</td>
<td>Use this module to determine the application scope context in which a script runs. Since one script can call another script it is possible to have multiple application scope context changes while running a series of scripts.</td>
</tr>
<tr>
<td>Enable Session Debug</td>
<td>Use this related link to enable the generation of log messages for a particular application. Application scripts that use GlideSystem logging methods will generate output to the log at the indicated verbosity level.</td>
</tr>
</tbody>
</table>

When multiple applications contribute to the debug output, the system adds a new section called **Apps** to the display a list of the applications writing to the session log. Clicking on the check box next to the application name hides or displays the application's associated debug messages.
Sample application debug output of business rules

Debugging scopes

Application developers can use the **Debug Scopes** module to display information about when the system switches between custom applications to run server-side scripts.

When enabled, the system displays a message whenever the system switches to a custom application to run a server-side script.
Sample debug scopes output from the incident table

Every time the system runs a server-side script object it enters the script's scope context. When the script finishes running, the script exits the scope context. The debugging messages track changes to the script scope context.

The debugging message displays a greater than character > each time the system enters a script object's context, and displays a less than character < every time the system exits a script object's context. In cases where one script calls another the debugging message adds another greater than character to the path for each call. For example, if a business rule calls a script include, which in turn calls another script object there would three characters in the path such as:

```
> Entering scope [x_app_one]
>> Entering scope [x_app_two]
>>> Entering scope [x_app_three]
```

Note: The system does not display entering or exiting messages for script objects in the global scope.

Application developers may want to enable other debugging options to in conjunction with this option to see information about the possible source of the server-side script such as Debug Business Rule.

Debugging business rules

Debugging business rules can be achieved with resources available in the ServiceNow product.
1. Tools

The first step in the process is to identify tools which will help you figure out what's wrong.

Debugging tools

<table>
<thead>
<tr>
<th>Debugging tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Dictionary</td>
<td>Navigate to System Definition &gt; Dictionary. The dictionary provides a list of all tables within your instance and can be invaluable when trying to locate information.</td>
</tr>
<tr>
<td>System Log</td>
<td>Navigate to System Logs &gt; System Log. You can place alert statements in your business rule which can write information to the log.</td>
</tr>
<tr>
<td>Debug Business Rule (Details)</td>
<td>Navigate to System Diagnostics &gt; Session Debug &gt; Debug Business Rule (Details). This debugging module displays the results business rules. Use this module to see if conditions are being met and values are being set as expected.</td>
</tr>
<tr>
<td>Alert Messages</td>
<td>There are several system functions that allow you to print messages to the page, the field or the log file. See Scripting alert, info, and error messages.</td>
</tr>
<tr>
<td>Business Rule Examples</td>
<td>Sometimes you can find what you're looking for in scripts others have written, including business rule error messages, or by building an OR query.</td>
</tr>
<tr>
<td>GlideRecord Information</td>
<td>This is the basic syntax used to query the database for information. See Using GlideRecord to query tables. GlideRecord also includes aggregation support.</td>
</tr>
</tbody>
</table>

2. Variables

The next step is to gain some insight into the behavior of your business rule. For every action except an insert, you will more than likely use a query to get your record(s).

```javascript
var rec = new GlideRecord('incident');
rec.addQuery('active',true);
rec.query();
while (rec.next()) {
    gs.print(rec.number + ' exists');
}
```

To verify whether your query is actually returning records you can use gs.addInfoMessage to display information at the top of the screen.

```javascript
var rec = new GlideRecord('incident');
rec.addQuery('active',true);
rec.query();
gs.addInfoMessage("This is rec.next: " + rec.next());
while (rec.next()) {
    gs.print(rec.number + ' exists');
}
```
If your query returns no records you see the following:

```
This is rec.next: false
```

Use this technique to verify every variable within your business rule contains expected values.

**Tip:** If necessary, break your script down into individual pieces and verify each piece works separate from the whole and then put them all back together one step at a time.

### 3. Locating information

The last step is to make sure you know where to find the information your rule is looking for.

In the ServiceNow application, one table can extend another table. This means when searching for information, you might need to query the parent table for the extended table’s sys_id to find what you seek.

A good example is the `sc_task` table, which extends the `task` table. The script below queries the extended table (`sc_task`) for the current `sys_id` and then query the parent table (`task`) for records with the matching `sys_id`, and then prints out the work notes field.

```javascript
var kids = new GlideRecord('sc_task');
kids.query();

gs.addInfoMessage("This is requested item number: " + current.number);
gs.print("This is the requested item number: " + current.number);

while (kids.next()) {
    var parents = new GlideRecord('task');
    parents.addQuery('sys_id', '=', kids.sys_id);
    parents.query();

    while(parents.next()) {
        gs.addInfoMessage("This is task number: " + parents.number);
        gs.print("This is task number: " + parents.number);
        gs.addInfoMessage("These are the work notes: " + parents.work_notes);
        gs.print("These are the work notes: " + parents.work_notes);
    }
}
```

### Debugging classifications

You must add a system property to enable classification debugging.

The resulting log entries list the name of each classifier that runs, along with all the names and values that are available to the criteria in the classifier. To log debugging information about classifications, add the following system property.
<table>
<thead>
<tr>
<th>System Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>glide.discovery.debug.classification</td>
<td>Enables debugging information for process classification.</td>
</tr>
<tr>
<td></td>
<td>· Type: true</td>
</tr>
<tr>
<td></td>
<td>· Default Value: false</td>
</tr>
<tr>
<td></td>
<td>· Location: Add to the System Properties (sys_properties) table</td>
</tr>
</tbody>
</table>

**Field watcher**

The field watcher tool tracks and displays all actions that the system performs on a selected form field.

Administrators can use the field watcher to figure out what happens to the field and how the value of the field changes when an event such as the firing of a business rule or enforcement of a data policy, takes place. Administrators can also impersonate non-admin users to debug what happens when those users make changes on an instance. Only one field can be watched at a time. Non-admin users with the impersonator role have access to the field watcher feature.

**How the field watcher works**

The Field Watcher tool logs activity when any of the following events occur on a field:

- The default value is set on the field.
- User access rights for the field change due to an ACL or dictionary setting.
- A data policy prevents the value from being set.
- A reference qualifier query of the field value executes.
- A UI policy changes a field to or from read-only, visible, mandatory, or editable.
- A dependent value in another field restricts field choices.
- The value of the field is set or changed based on:
  - Assignment rules
  - Actions from an engine, such as the workflow engine
  - Business rules
  - User entries
  - Client scripts
  - UI actions

**Note:** The field watcher works only on form fields. It cannot be used on list fields. Also, field watcher is not available on password-protected fields or encrypted fields. Field watcher is only available within the UI frame. The option to watch a field does not appear in the context menu if you open a record outside of the UI frame, for example, in a new tab.

**Use field watcher**

Access field-level debugging information using the field watcher.

1. Navigate to the form for which you want to view field-level debugging information.
2. Activate field watcher by right-clicking any field label on a form and select **Watch** - `<field name>`.
The debug icon ( ), appears next to the field label. From this point on, the field watcher records every action taken on the selected field. For example, if you are watching a Priority field, if the priority is changed from Moderate to Low and the record is updated, the field watcher will display information about that change.

3. View the field watcher log by clicking the debug icon.
A new pane opens at the bottom of the screen, showing a field watcher tab. It may also show tabs for JavaScript Logging and JavaScript Debugger.

4. Click the Field Watcher tab, if needed.
5. Stop watching a field by right-clicking the field and selecting Unwatch - <field name>. To watch another field, right-click that field and select Watch - <field name>.
6. Clear the field watcher log by clicking the clear log button ( ).
7. Resize the field watcher pane by dragging the splitter bar up or down. Dragging the splitter bar to the bottom of the screen closes the field watcher pane. Reopen the pane by clicking the debug icon again.

Field watcher tab details
The field watcher displays field information and configuration options.

The left-side of the Field Watcher tab shows the following information for the watched field.

- **Table**: table to which the field belongs.
- **Element**: field label.
- **Type**: type of data stored in the field.
On the right-side of the Field Watcher tab, select the types of activity information you want to see for the selected field. Clear the check box for any type of information that is not needed.

**Watching a hidden field**
Administrators may need to watch a hidden field.
1. Use the dictionary to determine the column name of the field.
2. Elevate privileges to the security_admin role.
3. Navigate to System Definition > Scripts Background.
4. In Run script (JavaScript executed on server), enter the following command:

   ```javascript
   gs.getSession (). setWatchField ( "hidden_field" );
   ```

   Replace hidden_field with the column name of the hidden field.
5. Navigate to the form containing the missing field.
   The Field Watcher tab output contains information about the hidden field.

**Viewing information for the watched field**
When information for a watched field is changed and the record is updated, the field watcher tab displays relevant information at the bottom.
- **Timestamp**: time the field was changed using the HH:MM:SS (ms) format.
- **Orange text**: server-side changes, such as ACLs.
- **Blue text**: client-side changes, such as client scripts.

- **Type of object that changed the field and its associated name**: The type of item that changed on the field; for example, **CLIENT SCRIPT**, **BUSINESS RULE**, or **ACL**. In the case of scripts, business rules, or other configuration-type fields, field watcher displays the name of the script or business rule that changed the field, if any. Click the name to go directly to the record for that item.
- **Old and new values**: The old and new values for the field, if the value changed. Field watcher does not record the value if it was inserted in the form by default at the time the record was created.
- **Additional information**: Call tracing information, such as the name of the script engine or workflow that changed the field. Click the plus icon to expand the selection.
  - **Orange text**: Indicates server-side activity.
  - **Blue text**: Indicates client-side activity.

**Example: Watching the incident priority**

The following example shows what happens to the **Priority** field on the incident form when both the **Impact** and **Urgency** fields change.

The incident form has two client-side data lookups change the priority. Additionally, server-side ACLs and the data lookup engine fire when the record is saved. Finally, a client-side UI policy sets the **Priority** field back to read-only, which is the default setting.

**Watching the incident priority**

<table>
<thead>
<tr>
<th>Original values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority: 1 - Critical</td>
</tr>
<tr>
<td>Impact: 1 - High</td>
</tr>
<tr>
<td>Urgency: 1 - High</td>
</tr>
</tbody>
</table>

**First Change**

1. The user changes the **Impact** value to 3 - Low.
2. The priority automatically changes to 3 - Moderate based on the **Priority Lookup** data lookup definition used by default in ServiceNow incidents.

**Note**: At this point, the record has not been saved.

**Second Change**

1. The user changes the **Urgency** value to 2 - Medium.
2. The priority automatically changes to 4 - Low based on the same **Priority Lookup** data lookup definition.
3. The user saves the record by right-clicking the form header and choosing **Save**.
Field watcher example

Note: The values that change from 1 to 3, and then from 3 to 4, refer to the numerical values in the choice list.

Tutorials

The following video demonstrates how to perform field-level debugging with the field watcher.

Writing to the debug log

To write to the debug log in your client-side JavaScript, or UI policies, make a call to the global function `jslog()`.

An example of using `jslog()` in JavaScript:

```javascript
function logData (r ) {
    lastLogDate  = r. responseXML. documentElement. getAttribute
    "last_log_entry" ) ; var items  = r. responseXML. getElementsByTagName
    "log" ) ;
    jslog ( "response=" + r. responseText ) ;
}
```

Additionally, when client scripts run, the name of the client script and timing information is displayed. This can be useful in determining which scripts are running and whether they are impacting performance.
Debug UI policies

Enabling the `glide.ui.ui_policy_debug` property lets you monitor the processing of UI actions.

Here are some sample log events from an incident policy that sets fields to read-only if the incident_state is closed.

```
GlideFieldPolicy: Evaluating condition
GlideFieldPolicy:     incident_state (7) = 7 -> true
GlideFieldPolicy:     --->>> TRUE
GlideFieldPolicy:    Setting opened_at disabled to true
GlideFieldPolicy:    Setting opened_by disabled to true
GlideFieldPolicy:    Setting closed_at disabled to true
GlideFieldPolicy:    Setting closed_by disabled to true
GlideFieldPolicy:    Setting company disabled to true
```

Access the JavaScript log

JavaScript that runs on the browser, such as client scripts, can include a call to `jslog()` to send information to the JavaScript log. Users with the admin role can access this log.

1. Open the JavaScript log by navigating to the appropriate location for your version of the UI.

<table>
<thead>
<tr>
<th>UI16</th>
<th>UI15</th>
<th>UI11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click the gear icon in the banner frame.</td>
<td>1. Click the gear icon in the banner frame.</td>
<td>Click the debug icon in the banner frame.</td>
</tr>
<tr>
<td>2. Click the Developer section.</td>
<td>2. Click JavaScript Log and Field Watcher.</td>
<td></td>
</tr>
<tr>
<td>3. Toggle the JavaScript Log and Field Watcher switch.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A new pane opens at the bottom of the screen. It shows the JavaScript Log tab and may also show the Field Watcher tab.

2. If needed, select the JavaScript Log tab.
3. Click the clear icon to clear the contents of the log, as needed.

JavaScript debug window

The JavaScript debug window appears in a bottom pane of the user interface when an administrator turns on debugging.

Use the debug window to access these tools.

- JavaScript Log: JavaScript that runs on the browser, such as client scripts, can include a call to `jslog()` to send information to the JavaScript log.
- Field Watcher: a tool that tracks and displays all actions that the system performs on a selected form field.
Using the JavaScript debug window

The JavaScript debug window enables access to the JavaScript Log and the Field Watcher tools. The steps to access the JavaScript debug window depend on which UI version you are using.

1. Open the JavaScript debug window by navigating to the appropriate location for your version of the UI.

   - **UI16**
     1. Click the gear icon in the banner frame.
     2. Click the **Developer** section.
     3. Toggle the **JavaScript Log and Field Watcher** switch.

   - **UI15**
     1. Click the gear icon in the banner frame.
     2. Click **JavaScript Log and Field Watcher**.

   - **UI11**
     Click the debug icon ( commodo ) in the banner frame.

The JavaScript debug window opens at the bottom of the screen. The tab that is currently active in the window is the last tab that was active when the window was closed.

2. Click a tab to use one of the debug window features.
   - JavaScript Log
   - Field Watcher

**JS Code Coverage Debug**

The JS Code Coverage Debug application allows administrators and application developers to log the scripts triggered during a user session and then review which lines of code the system ran.
Users with the js_coverage_debugger role can debug scripts without having to set breakpoints or review onscreen debug messages. Instead, the system saves script usage data in the JavaScript Code Coverage (sys_js_code_coverage) table. Each JavaScript Code Coverage record contains:

- The user session that called the script
- The script record the system called identified by table, sys_id, and script field
- The script record the system called identified by type and name
- The transaction that called the script
- The start time of the transaction
- The contents of the script field highlighted to indicate which lines the system ran

```
var build = gs.getProperty("glide.buildname");
if (GlideStringUtil.notNull(build))
    build = build.substring(0,1).toLowerCase();

var collision = new GlideRecord("sys_embedded_help_content");
collision.addActiveQuery();
collision.addQuery("page", current.page);
collision.addQuery("modifier", current.modifier);
collision.addQuery("product", current.product);
if (current.isValidRecord() && GlideStringUtil.notNull(current.sys_id))
    collision.addQuery("sys_id", "!=" current.sys_id);

if (GlideStringUtil.notNull(current.qualifier))
    collision.addQuery("qualifier", current.qualifier);
else
    collision.addNullQuery("qualifier");

if (current.version == "all" && GlideStringUtil.notNull(build))
    collision.addQuery("version", build).addOrCondition("version", "all");
else
    collision.addQuery("version", current.version);
collision.query();
if (collision.next()) {
    if (collision.canWrite())
        gs.addErrorMessage(gs.getMessage("Embedded Help Composite Key Validation"));
```

**Sample code coverage highlighting**

**JS Code Coverage highlighting**

The JS Code Coverage application highlights script fields to indicate whether the system ran or skipped each line.
Sample code highlighting

The color of the highlight indicates how the system evaluated the code line.

### Meaning of code highlighting

<table>
<thead>
<tr>
<th>Highlight color</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>This is an executable line of code that the system ran during the session.</td>
</tr>
<tr>
<td>Red</td>
<td>This is an executable line of code that the system skipped for some reason. The system may have skipped an executable line of code because the necessary script conditions were not met or because the script function was never called. You may want to use the Script Debugger to determine why the system skipped the line of executable code.</td>
</tr>
</tbody>
</table>
Administrators and application developers can use this information to conduct more targeted debugging activities such as using the Script Debugger to determine why script conditions are not being met.

### Activate JS Code Coverage Debug

You can activate the JS Code Coverage Debug plugin (com.glide.js.coverage) if you have the admin role.

**Role required:** admin

1. Navigate to **System Definition > Plugins**.
2. Find and click the plugin name.
3. On the System Plugin form, review the plugin details and then click the **Activate/Upgrade** related link.

   - If the plugin depends on other plugins, these plugins are listed along with their activation status.
   - If the plugin has optional features that depend on other plugins, those plugins are listed under **Some files will not be loaded because these plugins are inactive**. The optional features are not installed until the listed plugins are installed (before or after the installation of the current plugin).
4. Optional: If available, select the **Load demo data** check box.
   - Some plugins include demo data—Sample records that are designed to illustrate plugin features for common use cases. Loading demo data is a good practice when you first activate the plugin on a development or test instance.
   - You can also load demo data after the plugin is activated by clicking the **Load Demo Data Only** related link on the System Plugin form.
5. Click **Activate**.

To see the components the plugin installed, refresh the plugin form and select the **Plugin Files** related list.

### Debug with JS Code Coverage Debug

Use JS Code Coverage Debug to record a user session and then review which scripts and lines of code the system ran.

**Role required:** admin or js_coverage_debugger

1. Navigate to **JS Code Coverage Debug > Enable Coverage**.
   - The system logs which scripts and code lines the system runs as well as displays session debug messages in the JS Code Coverage namespace.
2. Navigate to the table or page whose logic you want to test. 
   For example, navigate to **Incident > Create New**.

3. Trigger the script or scripts you want to test.
   For example, create an incident with an associate CI item to test several business rules.

4. When you have completed testing, navigate to **JS Code Coverage Debug > Disable Coverage**.
   The system stops logging script and code lines run.

5. Navigate to **JS Code Coverage Debug > Coverage Data**.
   The system displays the list of coverage data associated with the current user session.
<table>
<thead>
<tr>
<th>Script Name</th>
<th>Script Reference</th>
<th>Transaction Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_script_include.d23429ec0a80d89806901958</td>
<td>Script Include: JSON</td>
<td>#3894</td>
</tr>
<tr>
<td>sys_script_include.d65776c050a0a000a99901958</td>
<td>Script Include: AbstractAjaxProcessor</td>
<td>#3921</td>
</tr>
<tr>
<td>sys_script.78e8b7e0a00707479138c73920a68</td>
<td>Business Rule: Update Parent Incident Count</td>
<td>#3921</td>
</tr>
<tr>
<td>sys_script.62a769ba0a65000a496392a383</td>
<td>Business Rule: user query</td>
<td>#3897</td>
</tr>
<tr>
<td>sys_script.6df7a579f00000016f7b23171995</td>
<td>Business Rule: Run SLAs</td>
<td>#3921</td>
</tr>
<tr>
<td>sys_script.28beda577070000012e35722f2d47</td>
<td>Business Rule: Caller Close</td>
<td>#3921</td>
</tr>
<tr>
<td>sys_script.72a1a40a0a7460e049f627769105</td>
<td>Business Rule: mark resolved</td>
<td>#3921</td>
</tr>
<tr>
<td>sys_script.0792b383dca0a905d73f217f4c08</td>
<td>Business Rule: Task Active State Management</td>
<td>#3921</td>
</tr>
<tr>
<td>sys_script.61c188b20a0acbb000105</td>
<td>Script Include: SysMessageRaj</td>
<td>#3888</td>
</tr>
<tr>
<td>sys_script.05e29a86c8da88009011d75b1616052</td>
<td>Business Rule: insert incident</td>
<td>#3921</td>
</tr>
<tr>
<td>sys_script.26d4637bca10000e65749246a0d74</td>
<td>Business Rule: Process SLAs</td>
<td>#3921</td>
</tr>
<tr>
<td>sys_script.3d5e14640b000000391b</td>
<td>Script Include: IncidentState</td>
<td>#3921</td>
</tr>
<tr>
<td>sys_script.47f97b5c61228401464a7a9929a</td>
<td>Business Rule: task closer</td>
<td>#3921</td>
</tr>
<tr>
<td>sys_script.7de297a80a8010640d4f7b</td>
<td>Script Include: SysRefList</td>
<td>#3894</td>
</tr>
<tr>
<td>sys_script.include.b32d228c08a0066006d00</td>
<td>Script Include: ArrayUtil</td>
<td>#3920</td>
</tr>
<tr>
<td>sys_script.2e5724ac0c1172a630c027c28f4f</td>
<td>Business Rule: Incident Create Knowledge</td>
<td>#3921</td>
</tr>
<tr>
<td>sys_script.2b295d1c08016419979e606173</td>
<td>Business Rule: incident query</td>
<td>#3922</td>
</tr>
</tbody>
</table>
6. Select the script or transaction you want to review.

**JavaScript Code Coverage fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script Name</td>
<td>Displays the script run by table name, sys_id value, and script field.</td>
</tr>
<tr>
<td>Script Reference</td>
<td>Displays the script run by script type and name.</td>
</tr>
<tr>
<td>Transaction</td>
<td>Displays the transaction that called the script by thread ID and URL.</td>
</tr>
</tbody>
</table>

For example, select the **Script Reference** Business Rule: incident events. The system displays the JS Code Coverage Debug record.
```javascript
if (current.operation() !== 'insert' && current.comments.changes) {
    gs.eventQueue("incident.commented", current, gs.getUserId(), gs.getUserName());
}

if (current.operation() === 'insert') {
    gs.eventQueue("incident.inserted", current, gs.getUserId(), gs.getUserName());
}

if (current.operation() === 'update') {
    gs.eventQueue("incident.updated", current, gs.getUserId(), gs.getUserName());
}

if (current.assigned_to() !== null && current.assigned_to.changes) {
    gs.eventQueue("incident.assigned", current,
        current.assigned_to().getDisplayName(), previous.assigned_to().getDisplayName());
}

if (current.assignment_group() !== null && current.assignment_group.changes) {
    gs.eventQueue("incident.assignment-group", current,
        current.assignment_group().getDisplayName(), previous.assignment_group().getDisplayName());
}

if (current.priority.changes() && current.priority !== null) {
    gs.eventQueue("incident.priority", current, current.priority, previous.priority);
}
```
7. Review the **Script** field to determine which lines of code the system ran.
   For example, the business rule added the incident.inserted event to the event queue.

You determine which lines of code the system ran.

Use the code coverage information to do more targeted debugging activities such as set breakpoints and review variable values with the Script Debugger.

**Packages Call Removal tool**

The Packages Call Removal Tool provides modules to identify fields that might contain scripts, find scripts that contain Packages calls to ServiceNow Java classes, and to examine proposed script changes that eliminate those Packages calls. You can substitute your own changes in place of the proposed changes.

Packages calls to ServiceNow Java classes will be prevented in a future release. The Packages Call Removal tool helps prepare your instance to use the new API and includes the following scripts and pages:

- The Find Packages Fields script scans scripts for Packages calls to ServiceNow Java classes.
- The Find Packages Calls script proposes changes that remove Packages calls or replace them with GlideScriptable names.
- The Packages Call Items page lists, and enables you to work on, proposed changes to scripts.

The tool is available to users with the administrator role.

The tool can generate errors as it tries to generate preferred, scriptable alternatives for Packages calls to ServiceNow Java classes.

**Note:** If you intend to capture and migrate the changes that result from running the Packages Call Removal Tool, you should first create an update set.

**Activate the Packages Call Removal Tool**

You must activate the Packages Call Removal Tool plugin to access the tool.

Role required: admin

For evaluation, you can activate the plugin for an application that requires a purchased subscription on a non-production instance. To activate the plugin on production instances, you must purchase the subscription. To purchase a subscription, contact your ServiceNow account manager. For details on purchasing a plugin, see **Purchase a plugin**.

Some plugins require activation by ServiceNow personnel. Request these plugins through the HI Customer Service System instead of activating them yourself. For details, see **Request a plugin**.

For plugins that you can activate yourself, continue with the following steps.

1. Navigate to **System Definition > Plugins**.
2. Find and click the plugin name.
3. On the System Plugin form, review the plugin details and then click the **Activate/Upgrade** related link.

   If the plugin depends on other plugins, these plugins are listed along with their activation status.

   If the plugin has optional features that depend on other plugins, those plugins are listed under **Some files will not be loaded because these plugins are inactive**. The optional features are not installed until the listed plugins are installed (before or after the installation of the current plugin).
4. Optional: If available, select the **Load demo data** check box.

   Some plugins include demo data—Sample records that are designed to illustrate plugin features for common use cases. Loading demo data is a good practice when you first activate the plugin on a development or test instance.

   You can also load demo data after the plugin is activated by clicking the **Load Demo Data Only** related link on the System Plugin form.

5. Click **Activate**.

**Find a Packages call**

After you run the Find Packages Fields script to define the list of fields to search for Packages calls, run the Find Packages Calls script to generate the list of fields with Packages calls to ServiceNow Java classes. The script also proposes changes.

1. Click **(3) Find Packages Calls (script)** to execute a script that searches the fields for packages calls and populates the Packages Call Items list with proposed changes.
Script completed: (3) Find Packages Calls (script)

*** Script: Searching Packages Call Fields for Packages calls in script, populating Packages Call Items
*** Script: Next, go to the 'Packages Call Items' module
*** Script: Script output below is only relevant to interested admins
*** Script:
** Script: Searching cim_asset_acquisition_method for Packages calls in script
** Script: Searching cim_license.assigned_condition for Packages calls in script
** Script: Searching cim_license.entitlement_condition for Packages calls in script
** Script: Searching cim_contract.terms_and_conditions for Packages calls in script
** Script: Searching catalog_ui_policy.catalog_conditions for Packages calls in script
** Script: Searching ci_identifier.script for Packages calls in script
** Script: Searching cim_contract_history.terms_and_conditions for Packages calls in script
** Script: Searching close_data_preserver.condition for Packages calls in script
** Script: Searching cmdb_baseline.condition for Packages calls in script
** Script: Searching cmdb_deprecation.script for Packages calls in script
** Script: Searching cmdb_we_license_calculation.calculation for Packages calls in script
** Script: Searching cim_notification_script.condition for Packages calls in script
** Script: Searching cim_service_provider.condition for Packages calls in script
** Script: Searching cim_relative_duration.script for Packages calls in script
** Script: Searching cm_schedule_condition.condition for Packages calls in script
** Script: Searching cm_schedule_page.client_script for Packages calls in script
** Script: Searching cm_schedule_page.server_script for Packages calls in script
** Script: Searching cm_timeline_page.condition for Packages calls in script
** Script: Searching cm_timeline_page.style.condition for Packages calls in script
** Script: Searching content_block.condition for Packages calls in script
** Script: Searching content_block_detail.script for Packages calls in script
** Script: Searching content_block langue.script for Packages calls in script
** Script: Searching content_page_rule.condition for Packages calls in script
** Script: Searching content_type.summary for Packages calls in script
** Script: Searching content_type_detail.script for Packages calls in script
** Script: Searching cim_contract_sla.condition for Packages calls in script
** Script: Searching cim_contract_sla.start_condition for Packages calls in script
** Script: Searching cim_contract_sla.end_condition for Packages calls in script
** Script: Searching diagrammer_action.condition for Packages calls in script
** Script: Searching diagrammer_action.script for Packages calls in script
** Script: Searching diagrammer_action_language.script for Packages calls in script
2. Click **Packages Call Items** to display the list of affected fields on the Packages Call Items page.

The Packages Call Items page lists the items with Packages calls, shows each item's current state, the table that contains the field, the affected record, the number of Packages calls contained in the field's script, and the number of errors that occurred when the proposed script was generated.
### ServiceNow Custom Business Applications

<table>
<thead>
<tr>
<th>State</th>
<th>Table</th>
<th>Field</th>
<th>Record</th>
<th>Packages Call Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed</td>
<td>content_type</td>
<td>summary</td>
<td>2</td>
<td>Content Type: Incident</td>
</tr>
<tr>
<td>Completed</td>
<td>content_type</td>
<td>summary</td>
<td>2</td>
<td>Content Type: prod_item</td>
</tr>
<tr>
<td>Completed</td>
<td>sys_trigger</td>
<td>script</td>
<td>2</td>
<td>Schedule Item: Count Knowledge Like</td>
</tr>
<tr>
<td>Completed</td>
<td>sys_trigger</td>
<td>script</td>
<td>2</td>
<td>Schedule Item: Recover Stock Events</td>
</tr>
<tr>
<td>Completed</td>
<td>sys_scripts</td>
<td>script</td>
<td>1</td>
<td>Business Rule: Incident Script</td>
</tr>
<tr>
<td>Proposed</td>
<td>content_block_programmatic</td>
<td>programmatic_content</td>
<td>1</td>
<td>Dynamic Content: Catalog Search Results</td>
</tr>
<tr>
<td>Proposed</td>
<td>content_type</td>
<td>detail</td>
<td>1</td>
<td>Content Type: DEFAULT</td>
</tr>
<tr>
<td>Proposed</td>
<td>content_type</td>
<td>summary</td>
<td>1</td>
<td>Content Type: sqivariate</td>
</tr>
<tr>
<td>Proposed</td>
<td>sys_auto_script</td>
<td>script</td>
<td>1</td>
<td>Scheduled Script Execution: Asynchronous Import Set Transformer</td>
</tr>
<tr>
<td>Proposed</td>
<td>sys_choice</td>
<td>label</td>
<td>1</td>
<td>Choice: sys_user</td>
</tr>
<tr>
<td>Proposed</td>
<td>sys_home</td>
<td>condition</td>
<td>1</td>
<td>Welcome Page Section: 36e1af4c7e090360e6bc090360e6bc0</td>
</tr>
<tr>
<td>Proposed</td>
<td>sys_script</td>
<td>include</td>
<td>1</td>
<td>Script Includes: Delivery?MatchNode</td>
</tr>
<tr>
<td>Proposed</td>
<td>sys_trigger</td>
<td>job_context</td>
<td>1</td>
<td>Schedule Item: Add Index Events: process</td>
</tr>
<tr>
<td>Proposed</td>
<td>sys_trigger</td>
<td>script</td>
<td>1</td>
<td>Schedule Item: LDAP Refresh</td>
</tr>
<tr>
<td>Proposed</td>
<td>sys_trigger</td>
<td>script</td>
<td>1</td>
<td>Schedule Item: LDAP Notify</td>
</tr>
<tr>
<td>Proposed</td>
<td>sys_trigger</td>
<td>script</td>
<td>1</td>
<td>Schedule Item: ECE Exchange Rate Load</td>
</tr>
</tbody>
</table>
3. On the Packages Call Items page, click a record for any of the listed items to open the form and revise as described in Replacing Packages Calls.

As you work through the list, the State of the items are updated and the items are grouped by State:

- **Proposed**: A proposed revision exists for the field.
- **Error** (red): One or more errors occurred when the proposed script was generated.
- **Rejected** (gray): A proposed change has been rejected.
- **Completed** (green): The field has been successfully revised to remove Packages calls to ServiceNow Java classes.
- **Canceled** (gray): Since the proposed change was generated, either the original script has been changed to no longer require modification, or the original record no longer exists.

---

**Remove Packages calls from scripts**

The Packages Call Items page lists, and enables you to work on, proposed changes to scripts.

The Packages Call Items page lists all Packages calls to ServiceNow Java classes. From the page, you can open any item in a form to examine the proposed changes. You can accept a proposed change as-is, or edit the code as necessary to remove the Packages calls.

Initially, all items without errors are in the Proposed state. When you take an action on a field, its state is updated. As you work through the list, each field is tracked by its current state: Proposed, Error, Rejected, Completed, or Canceled. You work through all of the fields until all of the Packages calls to ServiceNow Java classes have been removed.

1. To explore the details of an item in the list, click the link in the **State** column.
   The Package Migration Item form shows each line of the Original (current) version of the script and of the Proposed version. Lines with proposed changes are highlighted in yellow.
2. Work through the items until all are in the **Completed**, **Canceled**, or **Rejected** state.

<table>
<thead>
<tr>
<th>To accomplish this for an item</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accept the proposed change</strong></td>
<td>Click <strong>Execute Change</strong>.</td>
</tr>
<tr>
<td><strong>Reject the proposed change</strong></td>
<td>Click <strong>Reject Change</strong>. The change goes into the Rejected state. Rejecting a field allows you to defer action. To perform the change later, click <strong>Undo Rejection</strong>.</td>
</tr>
</tbody>
</table>
To accomplish this for an item | Do this
---|---
**Make the proposed change** | Click **Edit Proposed Version** to an editable script field where you can make changes. Below the edit field, a Diff pane displays line-by-line differences between the original and proposed versions.

After you save changes, code is highlighted in the Diff pane as follows:

- **Yellow** indicates code differences between the original version and the proposed version.
- **Green** indicates lines added to the proposed version that were not in the original version.
- **Red** indicates lines in the original version that have been removed from the proposed version.

1. Make changes in the **Proposed version** pane.

2. To save the change and rerun the comparison, click **Save**. If the change generates an error, ServiceNow puts the field into the **Error** state.

3. To revert a change, click the text in the **State** column to reopen the field on the Packages Call Items page, and then click **Revert Change**.

---

### Glide object name

The names of the new Glide objects that replace Packages calls are derived from the Java package name used in the Packages call.

Although the tool automatically substitutes the appropriate new scriptable name for each Packages call it encounters, in some circumstances it can be useful to know how to manually replace a Packages call with its new scriptable object equivalent. Use the steps below to determine the new script object name from the Packages call name. The replacement objects include the same methods and properties as the objects they replace.

To determine the new object name:

1. Note the third term in the Java package name. This is usually glide, but is sometimes snc or glideapp.

2. Drop all of the prefix terms, leaving only the last. For example, `Packages.com.glide.monitor.AbstractBucketCollector` becomes `AbstractBucketCollector`.


4. Verify that the name is valid by executing a `gs.print()` command in Scripts Background, specifying only the name with no quotes. For example:

   ```java
   gs.print( SncBaselineCMDB );
   ```

**Script:** `[JavaClass com.snc.cmdb.BaselineCMDB]`
Note that there are exceptions to this rule, such as "Glide", which replaces "Packages.com.glide.Glide", "TestExtension", which replaces "Packages.com.glide.junit.misc.TestExtension"; and "UINotification", which replaces "Packages.com.glide.ui.UINotification".

Glide object replacement list

This table lists the Glide classes and the Packages calls they replace.

---

#### GlideScriptable object replacement list

<table>
<thead>
<tr>
<th>GlideScriptable Class</th>
<th>Packages Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glide</td>
<td>Packages.com.glide.Glide</td>
</tr>
<tr>
<td>GlideAbstractDomainProvider</td>
<td>Packages.com.glide.db.domain.AbstractDomainProvider</td>
</tr>
<tr>
<td>GlideAbstractListener</td>
<td>Packages.com.glide.listener.AbstractListener</td>
</tr>
<tr>
<td>GlideActionManager</td>
<td>Packages.com.glide.ui.action.ActionManager</td>
</tr>
<tr>
<td>GlideAJAXScheduleItem</td>
<td>Packages.com.glide.schedules.AJAXScheduleItem</td>
</tr>
<tr>
<td>GlideAJAXSchedulePage</td>
<td>Packages.com.glide.schedules.AJAXSchedulePage</td>
</tr>
<tr>
<td>GlideAlertActions</td>
<td>Packages.com.glide.alerts.AlertActions</td>
</tr>
<tr>
<td>GlideappADSLoader</td>
<td>Packages.com.glideapp.ecc.ADSILoader</td>
</tr>
<tr>
<td>GlideappAJAXMapPage</td>
<td>Packages.com.glideapp.google_maps.AJAXMapPage</td>
</tr>
<tr>
<td>GlideappCalculationHelper</td>
<td>Packages.com.glideapp.servicecatalog.CalculationHelper</td>
</tr>
<tr>
<td>GlideappCart</td>
<td>Packages.com.glideapp.servicecatalog.Cart</td>
</tr>
<tr>
<td>GlideappCartItem</td>
<td>Packages.com.glideapp.servicecatalog.CartItem</td>
</tr>
<tr>
<td>GlideappCatalogCategoryBatcher</td>
<td>Packages.com.glideapp.servicecatalog.CategoryBatcher</td>
</tr>
<tr>
<td>GlideappCatalogItem</td>
<td>Packages.com.glideapp.servicecatalog.CategoryItem</td>
</tr>
<tr>
<td>GlideappCategory</td>
<td>Packages.com.glideapp.servicecatalog.Category</td>
</tr>
</tbody>
</table>

---

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>GlideScriptable Class</th>
<th>Packages Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlideappCategoryPopper</td>
<td>Packages.com.glideapp.servicecatalog.CategoryPopper</td>
</tr>
<tr>
<td>GlideappCattItemPopper</td>
<td>Packages.com.glideapp.servicecatalog.CattItemPopper</td>
</tr>
<tr>
<td>GlideappChartParameters</td>
<td>Packages.com.glideapp.chart.ChartParameters</td>
</tr>
<tr>
<td>GlideappChatRoom</td>
<td>Packages.com.glideapp.live.db.ChatRoom</td>
</tr>
<tr>
<td>GlideappCheckBoxQuestion</td>
<td>Packages.com.glideapp.questionset.CheckBoxQuestion</td>
</tr>
<tr>
<td>GlideappCMDBHelper</td>
<td>Packages.com.glideapp.ecc.CMDBHelper</td>
</tr>
<tr>
<td>GlideappCMDBSoftwareHelper</td>
<td>Packages.com.glideapp.ecc.CMDBSoftwareHelper</td>
</tr>
<tr>
<td>GlideappContainerAwareQuestionSet</td>
<td>Packages.com.glideapp.questionset.ContainerAwareQuestionSet</td>
</tr>
<tr>
<td>GlideappDateQuestion</td>
<td>Packages.com.glideapp.questionset.DateQuestion</td>
</tr>
<tr>
<td>GlideappDateTimeQuestion</td>
<td>Packages.com.glideapp.questionset.DateTimeQuestion</td>
</tr>
<tr>
<td>GlideappDeliveryPlan</td>
<td>Packages.com.glideapp.servicecatalog.DeliveryPlan</td>
</tr>
<tr>
<td>GlideappECCInputMessage</td>
<td>Packages.com.glideapp.ecc.ECCInputMessage</td>
</tr>
<tr>
<td>GlideappECCOutputMessage</td>
<td>Packages.com.glideapp.ecc.ECCOutputMessage</td>
</tr>
<tr>
<td>GlideappECCQueueConnector</td>
<td>Packages.com.glideapp.ecc.ECCQueueConnector</td>
</tr>
<tr>
<td>GlideappECCQueueProcessor</td>
<td>Packages.com.glideapp.ecc.ECCQueueProcessor</td>
</tr>
<tr>
<td>GlideappECCResponseMessage</td>
<td>Packages.com.glideapp.ecc.ECCResponseMessage</td>
</tr>
<tr>
<td>GlideappExpandableText</td>
<td>Packages.com.glideapp.live_feed.HTMLTransformers.ExpandableText</td>
</tr>
<tr>
<td>GlideappFixes</td>
<td>Packages.com.glideapp.servicecatalog.Fixes</td>
</tr>
<tr>
<td>GlideappHome</td>
<td>Packages.com.glideapp.home.Home</td>
</tr>
<tr>
<td>GlideappHomePage</td>
<td>Packages.com.glideapp.home.HomePage</td>
</tr>
<tr>
<td>GlideappIECC</td>
<td>Packages.com.glideapp.ecc.IECC</td>
</tr>
<tr>
<td>GlideappIOUpgrade</td>
<td>Packages.com.glideapp.servicecatalog.IOUpgrade</td>
</tr>
<tr>
<td>GlideappItemOptionsQuestionSet</td>
<td>Packages.com.glideapp.servicecatalog.ItemOptionsQuestionSet</td>
</tr>
<tr>
<td>GlideappJMSECCReceiver</td>
<td>Packages.com.glideapp.jms.JMSECCReceiver</td>
</tr>
<tr>
<td>GlideappJMSECCSender</td>
<td>Packages.com.glideapp.jms.JMSECCSender</td>
</tr>
<tr>
<td>GlideappKBIncludes</td>
<td>Packages.com.glideapp.knowledge.KBIncludes</td>
</tr>
<tr>
<td>GlideApplication</td>
<td>Packages.com.glide.sys.Application</td>
</tr>
<tr>
<td>GlideApplicationModule</td>
<td>Packages.com.glide.processors.ApplicationModule</td>
</tr>
<tr>
<td>GlideappListCollectorQuestion</td>
<td>Packages.com.glideapp.questionset.ListCollectorQuestion</td>
</tr>
<tr>
<td>GlideappLiveFeedEventHandler</td>
<td>Packages.com.glideapp.live_feed.LiveFeedEventHandler</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>GlideappLiveFeedUIAction</td>
<td>Packages.com.glideapp.live_feed.LiveFeedUIAction</td>
</tr>
<tr>
<td>GlideappLiveProfile</td>
<td>Packages.com.glideapp.live_common.LiveProfile</td>
</tr>
<tr>
<td>GlideappLiveUtils</td>
<td>Packages.com.glideapp.live.LiveUtils</td>
</tr>
<tr>
<td>GlideappLookupSelectQuestion</td>
<td>Packages.com.glideapp.questionset.LookupSelectQuestion</td>
</tr>
<tr>
<td>GlideappQuestion</td>
<td>Packages.com.glideapp.questionset.Question</td>
</tr>
<tr>
<td>GlideappQuestionChoice</td>
<td>Packages.com.glideapp.questionset.QuestionChoice</td>
</tr>
<tr>
<td>GlideappQueueHelper</td>
<td>Packages.com.glideapp.ecc.QueueHelper</td>
</tr>
<tr>
<td>GlideappQueueReader</td>
<td>Packages.com.glideapp.ecc.QueueReader</td>
</tr>
<tr>
<td>GlideappReferenceQuestion</td>
<td>Packages.com.glideapp.questionset.ReferenceQuestion</td>
</tr>
<tr>
<td>GlideappRequestItemWorkflow</td>
<td>Packages.com.glideapp.servicecatalog.RequestItemWorkflow</td>
</tr>
<tr>
<td>GlideappRequestNew</td>
<td>Packages.com.glideapp.servicecatalog.RequestNew</td>
</tr>
<tr>
<td>GlideappScriptHelper</td>
<td>Packages.com.glideapp.servicecatalog.ScriptHelper</td>
</tr>
<tr>
<td>GlideappSequencedQuestionSet</td>
<td>Packages.com.glideapp.questionset.SequencedQuestionSet</td>
</tr>
<tr>
<td>GlideappTaskApprovalHelper</td>
<td>Packages.com.glideapp.servicecatalog.TaskApprovalHelper</td>
</tr>
<tr>
<td>GlideappTimeAgo</td>
<td>Packages.com.glideapp.live_feed.TimeAgo</td>
</tr>
<tr>
<td>GlideappUpgradeQuestions</td>
<td>Packages.com.glideapp.survey.UpgradeQuestions</td>
</tr>
<tr>
<td>GlideappValveProcessor</td>
<td>Packages.com.glideapp.servicecatalog.valve.ValveProcessor</td>
</tr>
<tr>
<td>GlideappVariable</td>
<td>Packages.com.glideapp.servicecatalog.variables.Variable</td>
</tr>
<tr>
<td>GlideappVariablePoolQuestionSet</td>
<td>Packages.com.glideapp.servicecatalog.variables.VariablePoolQuestionSet</td>
</tr>
<tr>
<td>GlideappWizardIntercept</td>
<td>Packages.com.glideapp.wizard.WizardIntercept</td>
</tr>
<tr>
<td>GlideappWMILoader</td>
<td>Packages.com.glideapp.ecc.WMIloader</td>
</tr>
<tr>
<td>GlideappWorkflow</td>
<td>Packages.com.glideapp.workflow.Workflow</td>
</tr>
<tr>
<td>GlideappWorkflowHelper</td>
<td>Packages.com.glideapp.workflow.WorkflowHelper</td>
</tr>
<tr>
<td>GlideappYesNoQuestion</td>
<td>Packages.com.glideapp.questionset.YesNoQuestion</td>
</tr>
<tr>
<td>GlideAQueryExplanation</td>
<td>Packages.com.glide.db.explain.AQueryExplanation</td>
</tr>
<tr>
<td>GlideArchiver</td>
<td>Packages.com.glide.db.auxiliary.Archiver</td>
</tr>
<tr>
<td>GlideArchiveRecord</td>
<td>Packages.com.glide.db.auxiliary.ArchiveRecord</td>
</tr>
<tr>
<td>GlideArchiveRestore</td>
<td>Packages.com.glide.db.auxiliary.ArchiveRestore</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GlideArchiveStatus</td>
<td>Packages.com.glide.db.auxiliary.ArchiveStatus</td>
</tr>
<tr>
<td>GlideArchiveTable</td>
<td>Packages.com.glide.db.auxiliary.ArchiveTable</td>
</tr>
<tr>
<td>GlideARecurrence</td>
<td>Packages.com.glide.schedule.recurrence.ARecurrence</td>
</tr>
<tr>
<td>GlideAttachmentIndexDocument</td>
<td>Packages.com.glide.lucene.attachments.AttachmentIndexDocument</td>
</tr>
<tr>
<td>GlideAttachmentIndexTypes</td>
<td>Packages.com.glide.lucene.attachments.AttachmentIndexTypes</td>
</tr>
<tr>
<td>GlideAttributes</td>
<td>Packages.com.glide.util.GlideAttributes</td>
</tr>
<tr>
<td>GlideAuditDelete</td>
<td>Packages.com.glide.audit.AuditDelete</td>
</tr>
<tr>
<td>GlideAuditor</td>
<td>Packages.com.glide.script.Auditor</td>
</tr>
<tr>
<td>GlideAutomationEncrypter</td>
<td>Packages.com.glide.util.AutomationEncrypter</td>
</tr>
<tr>
<td>GlideBaseTag</td>
<td>Packages.com.glide.ui.jelly.tags.BaseTag</td>
</tr>
<tr>
<td>GlideBootstrap</td>
<td>Packages.com.glide.db.impex.Bootstrap</td>
</tr>
<tr>
<td>GlideBoundedIntProperty</td>
<td>Packages.com.glide.util.BoundedIntProperty</td>
</tr>
<tr>
<td>GlideCacheManager</td>
<td>Packages.com.glide.sys.cache.CacheManager</td>
</tr>
<tr>
<td>GlideCalendar</td>
<td>Packages.com.glide.schedule.GlideCalendar</td>
</tr>
<tr>
<td>GlideCalendarWeekEntry</td>
<td>Packages.com.glide.calendar.GlideCalendarWeekEntry</td>
</tr>
<tr>
<td>GlideCanceledUITransaction</td>
<td>Packages.com.glide.ui.CanceledUITransaction</td>
</tr>
<tr>
<td>GlideCascadeFromDelete</td>
<td>Packages.com.glide.db.CascadeFromDelete</td>
</tr>
<tr>
<td>GlideCatalogCloneWorker</td>
<td>Packages.com.glide.catalog.cloner.CatalogCloneWorker</td>
</tr>
<tr>
<td>GlideChannel</td>
<td>Packages.com.glide.channel.Channel</td>
</tr>
<tr>
<td>GlideChannelManager</td>
<td>Packages.com.glide.channel.ChannelManager</td>
</tr>
<tr>
<td>GlideChannelMessage</td>
<td>Packages.com.glide.channel.ChannelMessage</td>
</tr>
<tr>
<td>GlideChartFieldColors</td>
<td>Packages.com.glide.ui.chart.dataset.ChartFieldColors</td>
</tr>
<tr>
<td>GlideChartGeneratorFactory</td>
<td>Packages.com.glide.ui.chart.ChartGeneratorFactory</td>
</tr>
<tr>
<td>GlideChartUtil</td>
<td>Packages.com.glide.ui.chart.dataset.ChartUtil</td>
</tr>
<tr>
<td>GlideChartValue</td>
<td>Packages.com.glide.ui.chart.dataset.ChartValue</td>
</tr>
<tr>
<td>GlideChecksum</td>
<td>Packages.com.glide.util.Checksum</td>
</tr>
<tr>
<td>GlideChoice</td>
<td>Packages.com.glide.choice.Choice</td>
</tr>
<tr>
<td>GlideChoiceListGenerator</td>
<td>Packages.com.glide.choice.ChoiceListGenerator</td>
</tr>
<tr>
<td>GlideClientBrowserTimes</td>
<td>Packages.com.glide.client_transaction.ClientBrowserTimes</td>
</tr>
<tr>
<td>GlideClientNetworkTimes</td>
<td>Packages.com.glide.client_transaction.ClientNetworkTimes</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>GlideClusterState</td>
<td>Packages.com.glide.cluster.ClusterState</td>
</tr>
<tr>
<td>GlideClusterSynchronizer</td>
<td>Packages.com.glide.cluster.ClusterSynchronizer</td>
</tr>
<tr>
<td>GlideCMSLinkHelper</td>
<td>Packages.com.glide.cms.CMSLinkHelper</td>
</tr>
<tr>
<td>GlideCMSPageLink</td>
<td>Packages.com.glide.cms.CMSPageLink</td>
</tr>
<tr>
<td>GlideCollectionEnumerator</td>
<td>Packages.com.glide.utils.CollectionEnumerator</td>
</tr>
<tr>
<td>GlideCollectionQueryCalculator</td>
<td>Packages.com.glide.ui.CollectionQueryCalculator</td>
</tr>
<tr>
<td>GlideCollisionDetector</td>
<td>Packages.com.glide.update.collisions.CollisionDetector</td>
</tr>
<tr>
<td>GlideColumnAttributes</td>
<td>Packages.com.glide.db.impex.ColumnAttributes</td>
</tr>
<tr>
<td>GlideCompanyResolver</td>
<td>Packages.com.glide.misc.CompanyResolver</td>
</tr>
<tr>
<td>GlideCompiler</td>
<td>Packages.com.glide.script.Compiler</td>
</tr>
<tr>
<td>GlideCompositeElement</td>
<td>Packages.com.glide.db.CompositeElement</td>
</tr>
<tr>
<td>GlideConfiguration</td>
<td>Packages.com.glide.notification.Configuration</td>
</tr>
<tr>
<td>GlideContentConfig</td>
<td>Packages.com.glide.cms.ContentConfig</td>
</tr>
<tr>
<td>GlideContentPage</td>
<td>Packages.com.glide.cms.ContentPage</td>
</tr>
<tr>
<td>GlideContentSite</td>
<td>Packages.com.glide.cms.ContentSite</td>
</tr>
<tr>
<td>GlideContentType</td>
<td>Packages.com.glide.cms.ContentType</td>
</tr>
<tr>
<td>GlideContextMenu</td>
<td>Packages.com.glide.db_context_menu.ContextMenu</td>
</tr>
<tr>
<td>GlideContextMenuItem</td>
<td>Packages.com.glide.db_context_menu.ContextMenuItem</td>
</tr>
<tr>
<td>GlideController</td>
<td>Packages.com.glide.script.GlideController</td>
</tr>
<tr>
<td>GlideConverter</td>
<td>Packages.com.glide.currency.Converter</td>
</tr>
<tr>
<td>GlideCookieMan</td>
<td>Packages.com.glide.ui.CookieMan</td>
</tr>
<tr>
<td>GlideCounter</td>
<td>Packages.com.glide.utils.Counter</td>
</tr>
<tr>
<td>GlideCredentials</td>
<td>Packages.com.glide.communications.crypto.Credentials</td>
</tr>
<tr>
<td>GlideCryptoService</td>
<td>Packages.com.glide.security.CryptoService</td>
</tr>
<tr>
<td>GlideCSVExporter</td>
<td>Packages.com.glide.generators.CSVExporter</td>
</tr>
<tr>
<td>GlideDatabaseVerifier</td>
<td>Packages.com.glide.db.DatabaseVerifier</td>
</tr>
<tr>
<td>GlideDatabaseViewLink</td>
<td>Packages.com.glide.db_views.DatabaseViewLink</td>
</tr>
<tr>
<td>GlideDataSource</td>
<td>Packages.com.glide.db.impex.DataSource</td>
</tr>
<tr>
<td>GlideDate</td>
<td>Packages.com.glide.glideobject.GlideDate</td>
</tr>
<tr>
<td>GlideDateTime</td>
<td>Packages.com.glide.glideobject.GlideDateTime</td>
</tr>
<tr>
<td>GlideDateUtil</td>
<td>Packages.com.glide.utils.DateUtil</td>
</tr>
<tr>
<td>GlideDBAction</td>
<td>Packages.com.glide.db.DBACTION</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>GlideDBAggregateQuery</td>
<td>Packages.com.glide.db.DBAggregateQuery</td>
</tr>
<tr>
<td>GlideDBAggregateUtil</td>
<td>Packages.com.glide.db.DBAggregateUtil</td>
</tr>
<tr>
<td>GlideDBCategoryDebug</td>
<td>Packages.com.glide.secondary_db_pools.DBCategoryDebug</td>
</tr>
<tr>
<td>GlideDBChangeManager</td>
<td>Packages.com.glide.db.change.DBCategoryManager</td>
</tr>
<tr>
<td>GlideDBCompositeAction</td>
<td>Packages.com.glide.db.DBCompositeAction</td>
</tr>
<tr>
<td>GlideDBConfiguration</td>
<td>Packages.com.glide.db.DBConfiguration</td>
</tr>
<tr>
<td>GlideDBConfigurationManager</td>
<td>Packages.com.glide.db.DBConfigurationManager</td>
</tr>
<tr>
<td>GlideDBConfigurationManagerEventHandler</td>
<td>Packages.com.glide.db.DBConfigurationManagerEventHandler</td>
</tr>
<tr>
<td>GlideDBConfigurationParms</td>
<td>Packages.com.glide.db.DBConfigurationParms</td>
</tr>
<tr>
<td>GlideDBConfigurationV2Migrator</td>
<td>Packages.com.glide.db.DBConfigurationV2Migrator</td>
</tr>
<tr>
<td>GlideDBConnection</td>
<td>Packages.com.glide.db.pool.DBConnection</td>
</tr>
<tr>
<td>GlideDBConnectionPool</td>
<td>Packages.com.glide.db.pool.DBConectionPool</td>
</tr>
<tr>
<td>GlideDBConnectionPooler</td>
<td>Packages.com.glide.db.pool.DBConectionPooler</td>
</tr>
<tr>
<td>GlideDBDelete</td>
<td>Packages.com.glide.db.DBDelte</td>
</tr>
<tr>
<td>GlideDBImageProvider</td>
<td>Packages.com.glide.db_image.DBImageProvider</td>
</tr>
<tr>
<td>GlideDBMySQL</td>
<td>Packages.com.glide.db.rdbms.mysql.DBMySQL</td>
</tr>
<tr>
<td>GlideDBIndex</td>
<td>Packages.com.glide.db.DBIindex</td>
</tr>
<tr>
<td>GlideDBKeyStoreFactory</td>
<td>Packages.com.glide.certificates.DBKeyStoreFactory</td>
</tr>
<tr>
<td>GlideDBMacro</td>
<td>Packages.com.glide.ui.jelly.tags.form.DBMacro</td>
</tr>
<tr>
<td>GlideDBMicroStats</td>
<td>Packages.com.glide.db.DBMicroStats</td>
</tr>
<tr>
<td>GlideDBMultiTargetAction</td>
<td>Packages.com.glide.db.DBMultiTargetAction</td>
</tr>
<tr>
<td>GlideDBObjectManager</td>
<td>Packages.com.glide.db.DBOBJECTManager</td>
</tr>
<tr>
<td>GlideDBObjectToken</td>
<td>Packages.com.glide.db.DBOBJECTToken</td>
</tr>
<tr>
<td>GlideDBPoolTest</td>
<td>Packages.com.glide.secondary_db_pools.DBPooTest</td>
</tr>
<tr>
<td>GlideDBPropertiesConfig</td>
<td>Packages.com.glide.db.DBPropertiesConfig</td>
</tr>
<tr>
<td>GlideDBQuery</td>
<td>Packages.com.glide.db.DBQuery</td>
</tr>
<tr>
<td>GlideDBTypes</td>
<td>Packages.com.glide.db.DBTypes</td>
</tr>
<tr>
<td>GlideDBUpdate</td>
<td>Packages.com.glide.db.DBDelte</td>
</tr>
<tr>
<td>GlideDBUtil</td>
<td>Packages.com.glide.db.DBDelte</td>
</tr>
<tr>
<td>GlideDBView</td>
<td>Packages.com.glide.db.DBView</td>
</tr>
<tr>
<td>GlideDebugEvaluator</td>
<td>Packages.com.glide.jsdebug.DebugEvaluator</td>
</tr>
<tr>
<td>GlideDefaultUpdateSaver</td>
<td>Packages.com.glide.update.saver.DefaultUpdateSaver</td>
</tr>
<tr>
<td>GlideDiagram</td>
<td>Packages.com.glide.diagrammer.Diagram</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GlideDiagramEdge</td>
<td>Packages.com.glide.diagrammer.DiagramEdge</td>
</tr>
<tr>
<td>GlideDiagramElement</td>
<td>Packages.com.glide.diagrammer.DiagramElement</td>
</tr>
<tr>
<td>GlideDiagramNode</td>
<td>Packages.com.glide.diagrammer.DiagramNode</td>
</tr>
<tr>
<td>GlideDistUpgradeRunner</td>
<td>Packages.com.glide.dist.upgrade.runner.DistUpgradeRunner</td>
</tr>
<tr>
<td>GlideDomain</td>
<td>Packages.com.glide.db.domain.Domain</td>
</tr>
<tr>
<td>GlideDomainDisplay</td>
<td>Packages.com.glide.db.domain.DomainDisplay</td>
</tr>
<tr>
<td>GlideDomainHierarchy</td>
<td>Packages.com.glide.db.domain.DomainHierarchy</td>
</tr>
<tr>
<td>GlideDomainNumberProvider</td>
<td>Packages.com.glide.db.domain.DomainNumberProvider</td>
</tr>
<tr>
<td>GlideDomainPathDisplay</td>
<td>Packages.com.glide.db.domain.DomainPathDisplay</td>
</tr>
<tr>
<td>GlideDomainPathProvider</td>
<td>Packages.com.glide.db.domain.DomainPathProvider</td>
</tr>
<tr>
<td>GlideDomainSpoolProvider</td>
<td>Packages.com.glide.db.domain.DomainSpoolProvider</td>
</tr>
<tr>
<td>GlideDomainTree</td>
<td>Packages.com.glide.db.domain.DomainTree</td>
</tr>
<tr>
<td>GlideDomainUtil</td>
<td>Packages.com.glide.db.domain.DomainUtil</td>
</tr>
<tr>
<td>GlideDomainValidator</td>
<td>Packages.com.glide.db.domain.DOMainValidator</td>
</tr>
<tr>
<td>GlideDuration</td>
<td>Packages.com.glide.glideobject.GlideDuration</td>
</tr>
<tr>
<td>GlideECBDownloader</td>
<td>Packages.com.glide.currency.ECBDownloader</td>
</tr>
<tr>
<td>GlideECCQueueTransformer</td>
<td>Packages.com.glide.db.impex.ECCQueueTransformer</td>
</tr>
<tr>
<td>GlideElement</td>
<td>Packages.com.glide.script.GlideElement</td>
</tr>
<tr>
<td>GlideElementDescriptor</td>
<td>Packages.com.glide.db.ElementDescriptor</td>
</tr>
<tr>
<td>GlideElementIterator</td>
<td>Packages.com.glide.util.ElementIterator</td>
</tr>
<tr>
<td>GlideElementXMLSerializer</td>
<td>Packages.com.glide.script.GlideElementXMLSerializer</td>
</tr>
<tr>
<td>GlideEmail</td>
<td>Packages.com.glide.notification.Email</td>
</tr>
<tr>
<td>GlideEmailAction</td>
<td>Packages.com.glide.notification.outbound.EmailAction</td>
</tr>
<tr>
<td>GlideEmailFormatter</td>
<td>Packages.com.glide.notification.outbound.EmailFormatter</td>
</tr>
<tr>
<td>GlideEmailInbound</td>
<td>Packages.com.glide.notification.inbound.EmailInbound</td>
</tr>
<tr>
<td>GlideEmailOutbound</td>
<td>Packages.com.glide.notification.outbound.EmailOutbound</td>
</tr>
<tr>
<td>GlideEmailReader</td>
<td>Packages.com.glide.notification.inbound.EmailReader</td>
</tr>
<tr>
<td>GlideEmailSender</td>
<td>Packages.com.glide.notification.outbound.EmailSender</td>
</tr>
<tr>
<td>GlideEmailWatermark</td>
<td>Packages.com.glide.notification.EmailWatermark</td>
</tr>
<tr>
<td>GlideEmitter</td>
<td>Packages.com.glide.ui.jelly.Emitter</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>GlideEncrypter</td>
<td>Packages.com.glide.util.Encrypter</td>
</tr>
<tr>
<td>GlideEncryptionWrapperDB</td>
<td>Packages.com.glide.sys.security.EncryptionWrapperDB</td>
</tr>
<tr>
<td>GlideEncryptionWrapperDBAdmin</td>
<td>Packages.com.glide.sys.security.EncryptionWrapperDBAdmin</td>
</tr>
<tr>
<td>GlideEncryptionWrapperNAE</td>
<td>Packages.com.glide.sys.security.EncryptionWrapperNAE</td>
</tr>
<tr>
<td>GlideEncryptionWrapperNAEAdmin</td>
<td>Packages.com.glide.sys.security.EncryptionWrapperNAEAdmin</td>
</tr>
<tr>
<td>GlideEscalationManager</td>
<td>Packages.com.glide.escalation.EscalationManager</td>
</tr>
<tr>
<td>GlideEscalationTimerJobMarkII</td>
<td>Packages.com.glide.job.EscalationTimerJobMarkII</td>
</tr>
<tr>
<td>GlideEvaluator</td>
<td>Packages.com.glide.script.Evaluator</td>
</tr>
<tr>
<td>GlideEvent</td>
<td>Packages.com.glide.policy.Event</td>
</tr>
<tr>
<td>GlideEventManager</td>
<td>Packages.com.glide.policy.EventManager</td>
</tr>
<tr>
<td>GlideExcelExporter</td>
<td>Packages.com.glide.generators.ExcelExporter</td>
</tr>
<tr>
<td>GlideExcelLoader2</td>
<td>Packages.com.glide.db.impex.ExcelLoader2</td>
</tr>
<tr>
<td>GlideExecutionPlan</td>
<td>Packages.com.glide.execution_plan.ExecutionPlan</td>
</tr>
<tr>
<td>GlideExpressionWrapper</td>
<td>Packages.com.glide.ui.jelly.GlideExpressionWrapper</td>
</tr>
<tr>
<td>GlideExtensionPoint</td>
<td>Packages.com.glide.sys.ExtensionPoint</td>
</tr>
<tr>
<td>GlideFieldList</td>
<td>Packages.com.glide.processors.FieldList</td>
</tr>
<tr>
<td>GlideFile</td>
<td>Packages.com.glide.script.proxy.File</td>
</tr>
<tr>
<td>GlideFileUtil</td>
<td>Packages.com.glide.util.FileUtil</td>
</tr>
<tr>
<td>GlideFilter</td>
<td>Packages.com.glide.script.Filter</td>
</tr>
<tr>
<td>GlideFilterList</td>
<td>Packages.com.glide.script.FilterList</td>
</tr>
<tr>
<td>GlideFixCatalogPlans</td>
<td>Packages.com.glide.fixes.FixCatalogPlans</td>
</tr>
<tr>
<td>GlideFixDeliveryPlans</td>
<td>Packages.com.glide.fixes.FixDeliveryPlans</td>
</tr>
<tr>
<td>GlideFixGroups</td>
<td>Packages.com.glide Fixes.FixGroups</td>
</tr>
<tr>
<td>GlideFixItemOptionsAgain</td>
<td>Packages.com.glide Fixes.FixItemOptionsAgain</td>
</tr>
<tr>
<td>GlideFixRules</td>
<td>Packages.com.glide Fixes.FixRules</td>
</tr>
<tr>
<td>GlideFixSpellCheck</td>
<td>Packages.com.glide Fixes.FixSpellCheck</td>
</tr>
<tr>
<td>GlideFixStuff</td>
<td>Packages.com.glide Fixes.FixStuff</td>
</tr>
<tr>
<td>GlideFixUsers</td>
<td>Packages.com.glide Fixes.FixUsers</td>
</tr>
<tr>
<td>GlideForm</td>
<td>Packages.com.glide.ui.GlideForm</td>
</tr>
<tr>
<td>GlideFormCommon</td>
<td>Packages.com.glide.ui.GlideFormCommon</td>
</tr>
<tr>
<td>GlideFormulator</td>
<td>Packages.com.glide.ui.GlideFormulator</td>
</tr>
<tr>
<td>GlideGauge</td>
<td>Packages.com.glide.report.Gauge</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>GlideGovernor</td>
<td>Packages.com.glide.sys.util.Governor</td>
</tr>
<tr>
<td>GlideGregorianCalendar</td>
<td>Packages.com.glide.ui.jelly.tags.form.GroupByListTag</td>
</tr>
<tr>
<td>GlideGroup</td>
<td>Packages.com.glide.sys.Group</td>
</tr>
<tr>
<td>GlideGroupByListTag</td>
<td>Packages.com.glide.ui.jelly.tags.mergedata.HistoryTag2</td>
</tr>
<tr>
<td>GlideGuid</td>
<td>Packages.com.glide.util.Guid</td>
</tr>
<tr>
<td>GlideGovernor</td>
<td>Packages.com.glide.sys.util.Governor</td>
</tr>
<tr>
<td>GlideHTTPClient</td>
<td><a href="https://www.youtube.com/watch?v=9qYl_pBZej">https://www.youtube.com/watch?v=9qYl_pBZej</a></td>
</tr>
<tr>
<td>GlideHttpStatus</td>
<td>Packages.com.glide.communications.HTTPRequest</td>
</tr>
<tr>
<td>GlideHTTPResponse</td>
<td>Packages.com.glide.communications.HTTPResponse</td>
</tr>
<tr>
<td>GlideI18NStyle</td>
<td>Packages.com.glide.ui.I18NStyle</td>
</tr>
<tr>
<td>GlideICALUtil</td>
<td>Packages.com.glide.policy.ICALUtil</td>
</tr>
<tr>
<td>GlideConstants</td>
<td>Packages.com.glide.util.IConstants</td>
</tr>
<tr>
<td>GlideGlideRecord</td>
<td>Packages.com.glide.util.IGlideRecord</td>
</tr>
<tr>
<td>GlideImageLoader</td>
<td>Packages.com.glide.script.ImageLoader</td>
</tr>
<tr>
<td>GlideImpersonate</td>
<td>Packages.com.glide.sys.Impersonate</td>
</tr>
<tr>
<td>GlideImportLog</td>
<td>Packages.com.glide.db.impex.ImportLog</td>
</tr>
<tr>
<td>GlideImportMap</td>
<td>Packages.com.glide.db.impex.ImportMap</td>
</tr>
<tr>
<td>GlideImportMapField</td>
<td>Packages.com.glide.db.impex.ImportMapField</td>
</tr>
<tr>
<td>GlideImportSet</td>
<td>Packages.com.glide.system_import_set.ImportSet</td>
</tr>
<tr>
<td>GlideImportSetLoader</td>
<td>Packages.com.glide.system_import_set.ImportSetLoader</td>
</tr>
<tr>
<td>GlideImportSetRun</td>
<td>Packages.com.glide.system_import_set.ImportSetRun</td>
</tr>
<tr>
<td>GlideImportSetTransformer</td>
<td>Packages.com.glide.system_import_set.ImportSetTransformer</td>
</tr>
<tr>
<td>GlideImportSetTransformerWorker</td>
<td>Packages.com.glide.system_import_set.ImportSetTransformerWorker</td>
</tr>
<tr>
<td>GlideIndexDescriptor</td>
<td>Packages.com.glide.db.IndexDescriptor</td>
</tr>
<tr>
<td>GlideIndexUtils</td>
<td>Packages.com.glide.db.IndexUtils</td>
</tr>
<tr>
<td>GlideIntegerTime</td>
<td>Packages.com.glide.glideobject.IntegerTime</td>
</tr>
<tr>
<td>GlideInternalElementTypeChoiceList</td>
<td>Packages.com.glide.script.InternalElementTypeChoiceList</td>
</tr>
<tr>
<td>GlideInternalMonitor</td>
<td>Packages.com.glide.ui.monitor.InternalMonitor</td>
</tr>
<tr>
<td>GlideOMonitor</td>
<td>Packages.com.glide.db.io.IOStats</td>
</tr>
<tr>
<td>GlideIPAddressUtil</td>
<td>Packages.com.glide.util.IPAddressUtil</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>GlideIQueryCondition</td>
<td>Packages.com.glide.util.IQueryCondition</td>
</tr>
<tr>
<td>GlideIRow</td>
<td>Packages.com.glide.db.meta.IRow</td>
</tr>
<tr>
<td>GlideIRQuerySummary</td>
<td>Packages.com.glide.db.ir.IRQuerySummary</td>
</tr>
<tr>
<td>GlideIRQuerySummarySimple</td>
<td>Packages.com.glide.db.ir.IRQuerySummarySimple</td>
</tr>
<tr>
<td>GlideISecurityManager</td>
<td>Packages.com.glide.sys.security.ISecurityManager</td>
</tr>
<tr>
<td>GlideITableIterator</td>
<td>Packages.com.glide.db.access.ITableIterator</td>
</tr>
<tr>
<td>GlideJDBCLoader</td>
<td>Packages.com.glide.db.impex.JDBCLoader</td>
</tr>
<tr>
<td>GlideJDBCProbeTestWorker</td>
<td>Packages.com.glide.db.impex.JDBCProbeTestWorker</td>
</tr>
<tr>
<td>GlideJID</td>
<td>Packages.com.glide.xmpp.JID</td>
</tr>
<tr>
<td>GlideJSTestUtil</td>
<td>Packages.com.glide.autotester.JSTestUtil</td>
</tr>
<tr>
<td>GlideJSUtil</td>
<td>Packages.com.glide.script.JSUtil</td>
</tr>
<tr>
<td>GlideLabelEventHandler</td>
<td>Packages.com.glide.labels.LabelEventHandler</td>
</tr>
<tr>
<td>GlideLabelGenerator</td>
<td>Packages.com.glide.db.LabelGenerator</td>
</tr>
<tr>
<td>GlideLabelUtil</td>
<td>Packages.com.glide.labels.LabelUtil</td>
</tr>
<tr>
<td>GlideLDAP</td>
<td>Packages.com.glide.sys.ldap.LDAP</td>
</tr>
<tr>
<td>GlideLDAPConfig</td>
<td>Packages.com.glide.sys.ldap.LDAPConfig</td>
</tr>
<tr>
<td>GlideLDAPConfigurations</td>
<td>Packages.com.glide.sys.ldap.LDAPConfigurations</td>
</tr>
<tr>
<td>GlideLDAPErrorAnalyzer</td>
<td>Packages.com.glide.sys.ldap.LDAPErrorAnalyzer</td>
</tr>
<tr>
<td>GlideLDAPGroups</td>
<td>Packages.com.glide.sys.ldap.LDAPGroups</td>
</tr>
<tr>
<td>GlideLDAPRefresh</td>
<td>Packages.com.glide.sys.ldap.LDAPRefresh</td>
</tr>
<tr>
<td>GlideLDAPResult</td>
<td>Packages.com.glide.sys.ldap.LDAPResult</td>
</tr>
<tr>
<td>GlideLDAPTarget</td>
<td>Packages.com.glide.sys.ldap.LDAPTarget</td>
</tr>
<tr>
<td>GlideLDAPTransformQueue</td>
<td>Packages.com.glide.sys.ldap.LDAPTransformQueue</td>
</tr>
<tr>
<td>GlideLDAPUsers</td>
<td>Packages.com.glide.sys.ldap.LDAPUsers</td>
</tr>
<tr>
<td>GlideLDAPUserUpdate</td>
<td>Packages.com.glide.sys.ldap.LDAPUserUpdate</td>
</tr>
<tr>
<td>GlideList</td>
<td>Packages.com.glide.glideobject.GlideList</td>
</tr>
<tr>
<td>GlideListGroupProperties</td>
<td>Packages.com.glide.list_v2.ListGroupProperties</td>
</tr>
<tr>
<td>GlideListLabel</td>
<td>Packages.com.glide.ui.ListLabel</td>
</tr>
<tr>
<td>GlideListProperties</td>
<td>Packages.com.glide.list_v2.ListProperties</td>
</tr>
<tr>
<td>GlideListSearchQuery</td>
<td>Packages.com.glide.ui.ListSearchQuery</td>
</tr>
<tr>
<td>GlideLoader</td>
<td>Packages.com.glide.db.impex.Loader</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>GlideLoadTestDirector</td>
<td>Packages.com.glide.load_test.LoadTestDirector</td>
</tr>
<tr>
<td>GlideLocale</td>
<td>Packages.com.glide.sys.GlideLocale</td>
</tr>
<tr>
<td>GlideLocaleLoader</td>
<td>Packages.com.glide.currency.GlideLocaleLoader</td>
</tr>
<tr>
<td>GlideLock</td>
<td>Packages.com.glide.script.GlideLock</td>
</tr>
<tr>
<td>GlideLog</td>
<td>Packages.com.glide.util.GlideLog</td>
</tr>
<tr>
<td>GlideLogCleanup</td>
<td>Packages.com.glide.util.GlideLogCleanup</td>
</tr>
<tr>
<td>GlideLRUCache</td>
<td>Packages.com.glide.sys.cache.GlideLRUCache</td>
</tr>
<tr>
<td>GlideLuceneTextIndexEvent</td>
<td>Packages.com.glide.lucene.GlideLuceneTextIndexEvent</td>
</tr>
<tr>
<td>GlideMarkupWriter</td>
<td>Packages.com.glide.util.GlideMarkupWriter</td>
</tr>
<tr>
<td>GlideMemoryActive</td>
<td>Packages.com.glide.ui.monitor.GlideMemoryActive</td>
</tr>
<tr>
<td>GlideMemoryCache</td>
<td>Packages.com.glide.ui.monitor.GlideMemoryCache</td>
</tr>
<tr>
<td>GlideMemoryRecord</td>
<td>Packages.com.glide.script.GlideMemoryRecord</td>
</tr>
<tr>
<td>GlideMemorySwap</td>
<td>Packages.com.glide.ui.monitor.GlideMemorySwap</td>
</tr>
<tr>
<td>GlideMemoryTable</td>
<td>Packages.com.glide.ui.monitor.GlideMemoryTable</td>
</tr>
<tr>
<td>GlideMemoryTotal</td>
<td>Packages.com.glide.ui.monitor.GlideMemoryTotal</td>
</tr>
<tr>
<td>GlideMetaData</td>
<td>Packages.com.glide.script.GlideMetaData</td>
</tr>
<tr>
<td>GlideMobileExtensions</td>
<td>Packages.com.glide.ui.GlideMobileExtensions</td>
</tr>
<tr>
<td>GlideModule</td>
<td>Packages.com.glide.sys.GlideModule</td>
</tr>
<tr>
<td>GlideMultipleAction</td>
<td>Packages.com.glide.db.GlideMultipleAction</td>
</tr>
<tr>
<td>GlideMultipleDelete</td>
<td>Packages.com.glide.db.GlideMultipleDelete</td>
</tr>
<tr>
<td>GlideMultipleInsert</td>
<td>Packages.com.glide.db.GlideMultipleInsert</td>
</tr>
<tr>
<td>GlideMultipleUpdate</td>
<td>Packages.com.glide.db.GlideMultipleUpdate</td>
</tr>
<tr>
<td>GlideMutex</td>
<td>Packages.com.glide.sys.GlideMutex</td>
</tr>
<tr>
<td>GlideMySQLWatch</td>
<td>Packages.com.glide.sys.GlideMySQLWatch</td>
</tr>
<tr>
<td>GlideNumber</td>
<td>Packages.com.glide.script.GlideNumber</td>
</tr>
<tr>
<td>GlideNumberManager</td>
<td>Packages.com.glide.db.GlideNumberManager</td>
</tr>
<tr>
<td>GlideObjectManager</td>
<td>Packages.com.glide.glideobject.GlideObjectManager</td>
</tr>
<tr>
<td>GlideObjectUtil</td>
<td>Packages.com.glide.glideobject.GlideObjectUtil</td>
</tr>
<tr>
<td>GlideOrderingDefinitionCreator</td>
<td>Packages.com.glide.sorting.GlideOrderingDefinitionCreator</td>
</tr>
<tr>
<td>GlideOrderingManager</td>
<td>Packages.com.glide.sorting.GlideOrderingManager</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>GlideOverLoadedChoices</td>
<td>Packages.com.glide.script.OverLoadedChoices</td>
</tr>
<tr>
<td>GlidePivotTableSummaryTableWriter</td>
<td>Packages.com.glide.ui.chart.dataset.PivotTableSummaryTableWriter</td>
</tr>
<tr>
<td>GlidePlugin</td>
<td>Packages.com.glide.sys.Plugin</td>
</tr>
<tr>
<td>GlidePluginManager</td>
<td>Packages.com.glide.sys.PluginManager</td>
</tr>
<tr>
<td>GlidePluginManagerWorker</td>
<td>Packages.com.glide.sys.PluginManagerWorker</td>
</tr>
<tr>
<td>GlidePluginUtils</td>
<td>Packages.com.glide.sys.PluginUtils</td>
</tr>
<tr>
<td>GlidePOP3Reader</td>
<td>Packages.com.glide.notification.inbound.POP3Reader</td>
</tr>
<tr>
<td>GlidePOP3ReaderJob</td>
<td>Packages.com.glide.job.POP3ReaderJob</td>
</tr>
<tr>
<td>GlidePopup</td>
<td>Packages.com.glide.ui.Popup</td>
</tr>
<tr>
<td>GlidePriceGenerator</td>
<td>Packages.com.glide.currency.PriceGenerator</td>
</tr>
<tr>
<td>GlidePriceLoader</td>
<td>Packages.com.glide.currency.PriceLoader</td>
</tr>
<tr>
<td>GlideProcessor</td>
<td>Packages.com.glide.processors.Processor</td>
</tr>
<tr>
<td>GlideProgressMonitor</td>
<td>Packages.com.glide.worker.ProgressMonitor</td>
</tr>
<tr>
<td>GlideProgressWorker</td>
<td>Packages.com.glide.worker.ProgressWorker</td>
</tr>
<tr>
<td>GlideProperties</td>
<td>Packages.com.glide.util.GlideProperties</td>
</tr>
<tr>
<td>GlideProperty</td>
<td>Packages.com.glide.util.GlideProperty</td>
</tr>
<tr>
<td>GlidePublicPage</td>
<td>Packages.com.glide.ui.GlidePublicPage</td>
</tr>
<tr>
<td>GlideQueryBreadcrumbs</td>
<td>Packages.com.glide.misc.QueryBreadcrumbs</td>
</tr>
<tr>
<td>GlideQueryCondition</td>
<td>Packages.com.glide.db.conditions.QueryCondition</td>
</tr>
<tr>
<td>GlideQueryFormatter</td>
<td>Packages.com.glide.ui.jelly.tags.form.QueryFormatter</td>
</tr>
<tr>
<td>GlideQueryString</td>
<td>Packages.com.glide.db.QueryString</td>
</tr>
<tr>
<td>GlideQueryTerm</td>
<td>Packages.com.glide.db.QueryTerm</td>
</tr>
<tr>
<td>GlideRecord</td>
<td>Packages.com.glide.script.GlideRecord</td>
</tr>
<tr>
<td>GlideRecordCache</td>
<td>Packages.com.glide.sys.RecordCache</td>
</tr>
<tr>
<td>GlideRecordEnsurer</td>
<td>Packages.com.glide.misc.RecordEnsurer</td>
</tr>
<tr>
<td>GlideRecordFactory</td>
<td>Packages.com.glide.script.GlideRecordFactory</td>
</tr>
<tr>
<td>GlideRecordKeySetLoader</td>
<td>Packages.com.glide.script.GlideRecordKeySetLoader</td>
</tr>
<tr>
<td>GlideRecordLock</td>
<td>Packages.com.glide.script.GlideRecordLock</td>
</tr>
<tr>
<td>GlideRecordPopupGenerator</td>
<td>Packages.com.glide.calendar.RecordPopupGenerator</td>
</tr>
<tr>
<td>GlideRecordRollback</td>
<td>Packages.com.glide.script.GlideRecordRollback</td>
</tr>
<tr>
<td>GlideRecordSet</td>
<td>Packages.com.glide.script.GlideRecordSet</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>GlideRecordSimpleSerializer</td>
<td>Packages.com.glide.script.GlideRecordSimpleSerializer</td>
</tr>
<tr>
<td>GlideRecordXMLSerializer</td>
<td>Packages.com.glide.script.GlideRecordXMLSerializer</td>
</tr>
<tr>
<td>GlideReferenceField</td>
<td>Packages.com.glide.script.ReferenceField</td>
</tr>
<tr>
<td>GlideRegexUtil</td>
<td>Packages.com.glide.util.RegexUtil</td>
</tr>
<tr>
<td>GlideRegisterEscalationEvents</td>
<td>Packages.com.glide.fixes.RegisterEscalationEvents</td>
</tr>
<tr>
<td>GlideRelatedListReconciler</td>
<td>Packages.com.glide.misc.RelatedListReconciler</td>
</tr>
<tr>
<td>GlideRelationship</td>
<td>Packages.com.glide.sys.Relationship</td>
</tr>
<tr>
<td>GlideRelationships</td>
<td>Packages.com.glide.db.Relationships</td>
</tr>
<tr>
<td>GlideRelationshipUtil</td>
<td>Packages.com.glide.sys.RelationshipUtil</td>
</tr>
<tr>
<td>GlideRemoteGlideRecord</td>
<td>Packages.com.glide.communications.RemoteGlideRecord</td>
</tr>
<tr>
<td>GlideRenderProperties</td>
<td>Packages.com.glide.ui.RenderProperties</td>
</tr>
<tr>
<td>GlideReplaceUpdateFiles</td>
<td>Packages.com.glide.util.ReplaceUpdateFiles</td>
</tr>
<tr>
<td>GlideReplicationEngine</td>
<td>Packages.com.glide.replicator.ReplicationEngine</td>
</tr>
<tr>
<td>GlideRequestMap</td>
<td>Packages.com.glide.util.RequestMap</td>
</tr>
<tr>
<td>GlideRevertToOutOfBox</td>
<td>Packages.com.glide.update.RevertToOutOfBox</td>
</tr>
<tr>
<td>GlideRhinoHelper</td>
<td>Packages.com.glide.script.GlideRhinoHelper</td>
</tr>
<tr>
<td>GlideRhinoScope</td>
<td>Packages.com.glide.script.GlideRhinoScope</td>
</tr>
<tr>
<td>GlideRhinoScopeHandler</td>
<td>Packages.com.glide.script.GlideRhinoScopeHandler</td>
</tr>
<tr>
<td>GlideRhinoTestCase</td>
<td>Packages.com.glide.autotester.RhinoTestCase</td>
</tr>
<tr>
<td>GlideRRDBAlertProcessor</td>
<td>Packages.com.glide.rrdb.alerts.RRDBAlertProcessor</td>
</tr>
<tr>
<td>GlideRRDBDefinition</td>
<td>Packages.com.glide.rrdb.RRDBDefinition</td>
</tr>
<tr>
<td>GlideRunScriptJob</td>
<td>Packages.com.glide.job.RunScriptJob</td>
</tr>
<tr>
<td>GlideSchedule</td>
<td>Packages.com.glide.schedules.Schedule</td>
</tr>
<tr>
<td>GlideScheduleDateTime</td>
<td>Packages.com.glide.glideobject.ScheduleDateTime</td>
</tr>
<tr>
<td>GlideScheduleDateTimeSpan</td>
<td>Packages.com.glide.schedules.ScheduleDateTimeSpan</td>
</tr>
<tr>
<td>GlideScheduleItem</td>
<td>Packages.com.glide.schedules.ScheduleItem</td>
</tr>
<tr>
<td>GlideScheduler</td>
<td>Packages.com.glide.schedules.GlideScheduler</td>
</tr>
<tr>
<td>GlideScheduleTimeMap</td>
<td>Packages.com.glide.schedules.ScheduleTimeMap</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>GlideScriptedProgressWorker</td>
<td>Packages.com.glide.worker.ScriptedProgressWorker</td>
</tr>
<tr>
<td>GlideScriptEvaluator</td>
<td>Packages.com.glide.script.ScriptEvaluator</td>
</tr>
<tr>
<td>GlideScriptListener</td>
<td>Packages.com.glide.listener.ScriptListener</td>
</tr>
<tr>
<td>GlideScriptProcessor</td>
<td>Packages.com.glide.processors.ScriptProcessor</td>
</tr>
<tr>
<td>GlideScriptRecordUtil</td>
<td>Packages.com.glide.script.GlideRecordUtil</td>
</tr>
<tr>
<td>GlideScriptSystemUtilDB</td>
<td>Packages.com.glide.script.GlideSystemUtilDB</td>
</tr>
<tr>
<td>GlideScriptViewManager</td>
<td>Packages.com.glide.ui.ViewManager</td>
</tr>
<tr>
<td>GlideSearchQueryFormatter</td>
<td>Packages.com.glide.text_search.SearchQueryFormatter</td>
</tr>
<tr>
<td>GlideSelfCleaningMutex</td>
<td>Packages.com.glide.sys.lock.SelfCleaningMutex</td>
</tr>
<tr>
<td>GlideServiceAPIWrapper</td>
<td>Packages.com.glide.service_api.ServiceAPIWrapper</td>
</tr>
<tr>
<td>GlideServlet</td>
<td>Packages.com.glide.ui.GlideServlet</td>
</tr>
<tr>
<td>GlideServletRequest</td>
<td>Packages.com.glide.ui.GlideServletRequest</td>
</tr>
<tr>
<td>GlideServletStatus</td>
<td>Packages.com.glide.ui.GlideServletStatus</td>
</tr>
<tr>
<td>GlideSession</td>
<td>Packages.com.glide.sys.GlideSession</td>
</tr>
<tr>
<td>GlideSessionDebug</td>
<td>Packages.com.glide.sys.SessionDebug</td>
</tr>
<tr>
<td>GlideSessions</td>
<td>Packages.com.glide.ui.GlideSessions</td>
</tr>
<tr>
<td>GlideSessionSandbox</td>
<td>Packages.com.glide.script.GlideSessionSandbox</td>
</tr>
<tr>
<td>GlideShellCommand</td>
<td>Packages.com.glide.util.ShellCommand</td>
</tr>
<tr>
<td>GlideSimmerDown</td>
<td>Packages.com.glide.db.change.command.SimmerDown</td>
</tr>
<tr>
<td>GlideSimmerUp</td>
<td>Packages.com.glide.db.change.command.SimmerUp</td>
</tr>
<tr>
<td>GlideSimpleDateFormatEx</td>
<td>Packages.com.glide.util.SimpleDateFormatEx</td>
</tr>
<tr>
<td>GlideSimpleHttpClient</td>
<td>Packages.com.glide.communications.SimpleHttpClient</td>
</tr>
<tr>
<td>GlideSimpleScriptListener</td>
<td>Packages.com.glide.listener.SimpleScriptListener</td>
</tr>
<tr>
<td>GlideSMTPConnection</td>
<td>Packages.com.glide.notification.outbound.SMTPConnection</td>
</tr>
<tr>
<td>GlideSMTPSender</td>
<td>Packages.com.glide.notification.outbound.SMTPSender</td>
</tr>
<tr>
<td>GlideSMTPSenderJob</td>
<td>Packages.com.glide.job.SMTPSenderJob</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td>GlideSOAPDocument</td>
<td>Packages.com.glide.communications.soap.SOAPDocument</td>
</tr>
<tr>
<td>GlideSOAPRequest</td>
<td>Packages.com.glide.communications.soap.SOAPRequest</td>
</tr>
<tr>
<td>GlideSOAPResponse</td>
<td>Packages.com.glide.communications.soap.SOAPResponse</td>
</tr>
<tr>
<td>GlideSOAPSecurity</td>
<td>Packages.com.glide.processors.soap.SOAPSecurity</td>
</tr>
<tr>
<td>GlideSOAPSigner</td>
<td>Packages.com.glide.communications.soap.SOAPSigner</td>
</tr>
<tr>
<td>GlideSocket</td>
<td>Packages.com.glide.script.proxy.Socket</td>
</tr>
<tr>
<td>GlidesoftGlideAttributesImpl</td>
<td>Packages.com.glidesoft.util.xml.GlideAttributesImpl</td>
</tr>
<tr>
<td>GlidesoftXMLMemoryTable</td>
<td>Packages.com.glidesoft.util.xml.XMLMemoryTable</td>
</tr>
<tr>
<td>GlideSQLChildMonitor</td>
<td>Packages.com.glide.monitor.sql.SQLChildMonitor</td>
</tr>
<tr>
<td>GlideSQLDebug</td>
<td>Packages.com.glide.ui.diagnostics.SQLDebug</td>
</tr>
<tr>
<td>GlideSQLDeleteMonitor</td>
<td>Packages.com.glide.monitor.sql.SQLDeleteMonitor</td>
</tr>
<tr>
<td>GlideSQLInsertMonitor</td>
<td>Packages.com.glide.monitor.sql.SQLInsertMonitor</td>
</tr>
<tr>
<td>GlideSQLResponseMonitor</td>
<td>Packages.com.glide.monitor.sql.SQLResponseMonitor</td>
</tr>
<tr>
<td>GlideSQLSelectMonitor</td>
<td>Packages.com.glide.monitor.sql.SQLSelectMonitor</td>
</tr>
<tr>
<td>GlideSQLUpdateMonitor</td>
<td>Packages.com.glide.monitor.sql.SQLUpdateMonitor</td>
</tr>
<tr>
<td>GlideSSHCClient</td>
<td>Packages.com.glide.communications.SSHClient</td>
</tr>
<tr>
<td>GlideStack</td>
<td>Packages.com.glide.sys.GlideStack</td>
</tr>
<tr>
<td>GlideStatistician</td>
<td>Packages.com.glide.sys.stats.Statistician</td>
</tr>
<tr>
<td>GlideStatsInfo</td>
<td>Packages.com.glide.monitor_STATS_INFO</td>
</tr>
<tr>
<td>GlideStatus</td>
<td>Packages.com.glide.util.GlideStatus</td>
</tr>
<tr>
<td>GlideStopWatch</td>
<td>Packages.com.glide.util.StopWatch</td>
</tr>
<tr>
<td>GlideStorageUtils</td>
<td>Packages.com.glide.db.meta.StorageUtils</td>
</tr>
<tr>
<td>GlideStringCache</td>
<td>Packages.com.glide.sys.cache.StringCache</td>
</tr>
<tr>
<td>GlideStringInputStream</td>
<td>Packages.com.glide.util.StringInputStream</td>
</tr>
<tr>
<td>GlideStringList</td>
<td>Packages.com.glide.collections.StringList</td>
</tr>
<tr>
<td>GlideStringUtil</td>
<td>Packages.com.glide.util.StringUtil</td>
</tr>
<tr>
<td>GlideSubQuery</td>
<td>Packages.com.glide.db.conditions.SubQuery</td>
</tr>
<tr>
<td>GlideSubstituteURL</td>
<td>Packages.com.glide.notification.substitution.SubstituteURL</td>
</tr>
<tr>
<td>GlideSummaryTableGroupReader</td>
<td>Packages.com.glide.ui.chart.dataset.SummaryTableGroupReader</td>
</tr>
<tr>
<td>GlideSummaryTableOrderedReader</td>
<td>Packages.com.glide.ui.chart.dataset.SummaryTableOrderedReader</td>
</tr>
<tr>
<td>GlideSummaryTableReader</td>
<td>Packages.com.glide.ui.chart.dataset.SummaryTableReader</td>
</tr>
<tr>
<td>GlideSummaryTableWriter</td>
<td>Packages.com.glide.ui.chart.dataset.SummaryTableWriter</td>
</tr>
<tr>
<td>GlideSynchronizedLRUCache</td>
<td>Packages.com.glide.sys.cache.SynchronizedLRUCache</td>
</tr>
<tr>
<td>GlideSysAttachment</td>
<td>Packages.com.glide.ui.SysAttachment</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>GlideSysAttachmentInputStream</td>
<td>Packages.com.glide.ui.SysAttachmentInputStream</td>
</tr>
<tr>
<td>GlideSysBRThreadMonitor</td>
<td>Packages.com.glide.monitor.threads.SysBRThreadMonitor</td>
</tr>
<tr>
<td>GlideSysChoice</td>
<td>Packages.com.glide.script.SysChoice</td>
</tr>
<tr>
<td>GlideSysConcurrencyMonitor</td>
<td>Packages.com.glide.monitor.threads.SysConcurrencyMonitor</td>
</tr>
<tr>
<td>GlideSysCPUThreadMonitor</td>
<td>Packages.com.glide.monitor.threads.SysCPUThreadMonitor</td>
</tr>
<tr>
<td>GlideSysDateUtil</td>
<td>Packages.com.glide.sys.util.SysDateUtil</td>
</tr>
<tr>
<td>GlideSysDBThreadMonitor</td>
<td>Packages.com.glide.monitor.threads.SysDBThreadMonitor</td>
</tr>
<tr>
<td>GlideSysField</td>
<td>Packages.com.glide.db.SysField</td>
</tr>
<tr>
<td>GlideSysForm</td>
<td>Packages.com.glide.ui.SysForm</td>
</tr>
<tr>
<td>GlideSysForms</td>
<td>Packages.com.glide.ui.SysForms</td>
</tr>
<tr>
<td>GlideSysList</td>
<td>Packages.com.glide.ui.SysList</td>
</tr>
<tr>
<td>GlideSysListControl</td>
<td>Packages.com.glide.ui.SysListControl</td>
</tr>
<tr>
<td>GlideSysLog</td>
<td>Packages.com.glide.sys.SysLog</td>
</tr>
<tr>
<td>GlideSYSMany2Many</td>
<td>Packages.com.glide.db.SYSMany2Many</td>
</tr>
<tr>
<td>GlideSysSection</td>
<td>Packages.com.glide.ui.SysSection</td>
</tr>
<tr>
<td>GlideSysSemaphore</td>
<td>Packages.com.glide.sys.util.SysSemaphore</td>
</tr>
<tr>
<td>GlideSystem</td>
<td>Packages.com.glide.script.GlideSystem</td>
</tr>
<tr>
<td>GlideSystemDateUtil</td>
<td>Packages.com.glide.script.system.GlideSystemDateUtil</td>
</tr>
<tr>
<td>GlideSystemUtil</td>
<td>Packages.com.glide.util.SystemUtil</td>
</tr>
<tr>
<td>GlideSystemUtilDB</td>
<td>Packages.com.glide.script.system.GlideSystemUtilDB</td>
</tr>
<tr>
<td>GlideSystemUtilScript</td>
<td>Packages.com.glide.script.system.GlideSystemUtilScript</td>
</tr>
<tr>
<td>GlideSysThreadMonitor</td>
<td>Packages.com.glide.monitor.threads.SysThreadMonitor</td>
</tr>
<tr>
<td>GlideTable</td>
<td>Packages.com.glide.db.meta.Table</td>
</tr>
<tr>
<td>GlideTableChoiceList</td>
<td>Packages.com.glide.script.TableChoiceList</td>
</tr>
<tr>
<td>GlideTableCleaner</td>
<td>Packages.com.glide.misc.TableCleaner</td>
</tr>
<tr>
<td>GlideTableCleanerJob</td>
<td>Packages.com.glide.job.TableCleanerJob</td>
</tr>
<tr>
<td>GlideTableCreator</td>
<td>Packages.com.glide.db.impex.TableCreator</td>
</tr>
<tr>
<td>GlideTableDescriptor</td>
<td>Packages.com.glide.db.TableDescriptor</td>
</tr>
<tr>
<td>GlideTableGroupMover</td>
<td>Packages.com.glide.db.auxiliary.TableGroupMover</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>GlideTableManager</td>
<td>Packages.com.glide.db.TableManager</td>
</tr>
<tr>
<td>GlideTableMover</td>
<td>Packages.com.glide.db.auxiliary.TableMover</td>
</tr>
<tr>
<td>GlideTableParentChange</td>
<td>Packages.com.glide.db.table.TableParentChange</td>
</tr>
<tr>
<td>GlideTableParentColumnChange</td>
<td>Packages.com.glide.db.table.TableParentColumnChange</td>
</tr>
<tr>
<td>GlideTaskToken</td>
<td>Packages.com.glide.execution_plan.TaskToken</td>
</tr>
<tr>
<td>GlideTemplate</td>
<td>Packages.com.glide.script.Template</td>
</tr>
<tr>
<td>GlideTestAgent</td>
<td>Packages.com.glide.autotester.GlideTestAgent</td>
</tr>
<tr>
<td>GlideTextIndexEvent</td>
<td>Packages.com.glide.ts.event.TextIndexEvent</td>
</tr>
<tr>
<td>GlideThreadAttributes</td>
<td>Packages.com.glide.ui.GlideThreadAttributes</td>
</tr>
<tr>
<td>GlideThreadUtil</td>
<td>Packages.com.glide.ui.GlideThreadUtil</td>
</tr>
<tr>
<td>GlideTime</td>
<td>Packages.com.glide.glideobject.GlideTime</td>
</tr>
<tr>
<td>GlideTimelineFrameSeparator</td>
<td>Packages.com.glide.schedules.TimelineFrameSeparator</td>
</tr>
<tr>
<td>GlideTimelineItem</td>
<td>Packages.com.glide.schedules.TimelineItem</td>
</tr>
<tr>
<td>GlideTimelineSpan</td>
<td>Packages.com.glide.schedules.TimelineSpan</td>
</tr>
<tr>
<td>GlideTomcatHelper</td>
<td>Packages.com.glide.startup.TomcatHelper</td>
</tr>
<tr>
<td>GlideTransaction</td>
<td>Packages.com.glide.sys.Transaction</td>
</tr>
<tr>
<td>GlideTransactionManager</td>
<td>Packages.com.glide.sys.TransactionManager</td>
</tr>
<tr>
<td>GlideTransferAuditDataHelper</td>
<td>Packages.com.glide.audit.TransactionAuditDataHelper</td>
</tr>
<tr>
<td>GlideTransformer</td>
<td>Packages.com.glide.db.impex.transformer.Transformer</td>
</tr>
<tr>
<td>GlideTreePicker</td>
<td>Packages.com.glide.ui.TreePicker</td>
</tr>
<tr>
<td>GlideTSAnalyzer</td>
<td>Packages.com.glide.ts.cluster.TSAnalyzer</td>
</tr>
<tr>
<td>GlideTSAnalyticsProcessor</td>
<td>Packages.com.glide.ts.cluster.TSAnalyticsProcessor</td>
</tr>
<tr>
<td>GlideTSChainsHandler</td>
<td>Packages.com.glide.ts.trends.TSChainsHandler</td>
</tr>
<tr>
<td>GlideTSChainsLoader</td>
<td>Packages.com.glide.ts.trends.TSChainsLoader</td>
</tr>
<tr>
<td>GlideTSChainsPusher</td>
<td>Packages.com.glide.ts.trends.TSChainsPusher</td>
</tr>
<tr>
<td>GlideTSChainsSummarizer</td>
<td>Packages.com.glide.ts.trends.TSChainsSummarizer</td>
</tr>
<tr>
<td>GlideTSClusterDefinitions</td>
<td>Packages.com.glide.ts.cluster.TSClusterDefinitions</td>
</tr>
<tr>
<td>GlideTSDebug</td>
<td>Packages.com.glide.ts.util.TSDebug</td>
</tr>
<tr>
<td>GlideTSDidYouMean</td>
<td>Packages.com.glide.ts.util.TSDidYouMean</td>
</tr>
<tr>
<td>GlideTSGlobalKeywordSummarizer</td>
<td>Packages.com.glide.ts.trends.TSGlobalKeywordSummarizer</td>
</tr>
<tr>
<td>GlideTSIndexStatistician</td>
<td>Packages.com.glide.ts.stats.TSIndexStatistician</td>
</tr>
<tr>
<td>GlideTSIndexStopGenerator</td>
<td>Packages.com.glide.ts.stats.TSIndexStopGenerator</td>
</tr>
<tr>
<td>GlideTSIndexTables</td>
<td>Packages.com.glide.ts.indexer.TSIndexTables</td>
</tr>
<tr>
<td>GlideTSKeywordHandler</td>
<td>Packages.com.glide.ts.trends.TSKeywordHandler</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>GlideTSKeywordLoader</td>
<td>Packages.com.glide.ts.trends.TSKeywordLoader</td>
</tr>
<tr>
<td>GlideTSKeywordPusher</td>
<td>Packages.com.glide.ts.trends.TSKeywordPusher</td>
</tr>
<tr>
<td>GlideTSMoversViewer</td>
<td>Packages.com.glide.ts.cluster.TSMoversViewer</td>
</tr>
<tr>
<td>GlideTSSearchStatistician</td>
<td>Packages.com.glide.ts.stats.TSSearchStatistician</td>
</tr>
<tr>
<td>GlideTSSearchSummary</td>
<td>Packages.com.glide.ts.trends.TSSearchSummary</td>
</tr>
<tr>
<td>GlideTSTopSearches</td>
<td>Packages.com.glide.ts.util.TSTopSearches</td>
</tr>
<tr>
<td>GlideTSUtil</td>
<td>Packages.com.glide.ts.util.TSUtil</td>
</tr>
<tr>
<td>GlideTSVersion</td>
<td>Packages.com.glide.ts.TSVersion</td>
</tr>
<tr>
<td>GlideUIAction</td>
<td>Packages.com.glide.ui.action.UIAction</td>
</tr>
<tr>
<td>GlideUISession</td>
<td>Packages.com.glide.ui.Session</td>
</tr>
<tr>
<td>GlideUnloader</td>
<td>Packages.com.glide.db.impex.Unloader</td>
</tr>
<tr>
<td>GlideUpdateManager2</td>
<td>Packages.com.glide.update.UpdateManager2</td>
</tr>
<tr>
<td>GlideUpdateSet</td>
<td>Packages.com.glide.update.UpdateSet</td>
</tr>
<tr>
<td>GlideUpdateSetController</td>
<td>Packages.com.glide.system_update_set.UpdateSetController</td>
</tr>
<tr>
<td>GlideUpdateSetPreviewer</td>
<td>Packages.com.glide.update.UpdateSetPreviewer</td>
</tr>
<tr>
<td>GlideUpdateSetWorker</td>
<td>Packages.com.glide.update.UpdateSetWorker</td>
</tr>
<tr>
<td>GlideUpdateSyncher</td>
<td>Packages.com.glide.policy.UpdateSyncher</td>
</tr>
<tr>
<td>GlideUpdateTableChoiceList</td>
<td>Packages.com.glide.script.UpdateTableChoiceList</td>
</tr>
<tr>
<td>GlideUpgrade</td>
<td>Packages.com.glide.sys.Upgrade</td>
</tr>
<tr>
<td>GlideUpgradeArtifactManager</td>
<td>Packages.com.glide.misc.UpgradeArtifactManager</td>
</tr>
<tr>
<td>GlideUpgradeLog</td>
<td>Packages.com.glide.update.UpdateLog</td>
</tr>
<tr>
<td>GlideURI</td>
<td>Packages.com.glide.ui.GlideURI</td>
</tr>
<tr>
<td>GlideURL</td>
<td>Packages.com.glide.util.GlideURL</td>
</tr>
<tr>
<td>GlideURLUTF8Encoder</td>
<td>Packages.com.glide.util.URLUTF8Encoder</td>
</tr>
<tr>
<td>GlideURLUtil</td>
<td>Packages.com.glide.util.URLUtil</td>
</tr>
<tr>
<td>GlideUser</td>
<td>Packages.com.glide.sys.User</td>
</tr>
<tr>
<td>GlideUserGroup</td>
<td>Packages.com.glide.sys.UserGroup</td>
</tr>
<tr>
<td>GlideUtil</td>
<td>Packages.com.glide.util.GlideUtil</td>
</tr>
<tr>
<td>GlideWarDeleter</td>
<td>Packages.com.glide.misc.WarDeleter</td>
</tr>
<tr>
<td>GlideWarDownloader</td>
<td>Packages.com.glide.misc.WarDownloader</td>
</tr>
<tr>
<td>GlideWhiteListManager</td>
<td>Packages.com.glide.script.api.GlideWhiteListManager</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>GlideWikiModel</td>
<td>Packages.com.glide.wiki.GlideWikiModel</td>
</tr>
<tr>
<td>GlideWorkerThread</td>
<td>Packages.com.glide.worker.WorkerThread</td>
</tr>
<tr>
<td>GlideWorkerThreadManager</td>
<td>Packages.com.glide.sys.WorkerThreadManager</td>
</tr>
<tr>
<td>GlideWSClient</td>
<td>Packages.com.glide.util.WSClient</td>
</tr>
<tr>
<td>GlideWSDefinition</td>
<td>Packages.com.glide.wsdreader.WSDefinition</td>
</tr>
<tr>
<td>GlideWSDLReader</td>
<td>Packages.com.glide.wsdreader.GlideWSDLReader</td>
</tr>
<tr>
<td>GlideXMLChoiceListSerializer</td>
<td>Packages.com.glide.choice.XMLChoiceListSerializer</td>
</tr>
<tr>
<td>GlideXMLDocument</td>
<td>Packages.com.glide.util.XMLDocument</td>
</tr>
<tr>
<td>GlideXMLElementIterator</td>
<td>Packages.com.glide.util.XMLElementIterator</td>
</tr>
<tr>
<td>GlideXMLGlideRecordSerializer</td>
<td>Packages.com.glide.processors.xmlhttp.XMLGlideRecordSerializer</td>
</tr>
<tr>
<td>GlideXMLParameters</td>
<td>Packages.com.glide.util.XMLParameters</td>
</tr>
<tr>
<td>GlideXMLStats</td>
<td>Packages.com.glide.ui.XMLStats</td>
</tr>
<tr>
<td>GlideXMLSysMetaSerializer</td>
<td>Packages.com.glide.processors.xmlhttp.XMLSysMetaSerializer</td>
</tr>
<tr>
<td>GlideXMLUtil</td>
<td>Packages.com.glide.util.XMLUtil</td>
</tr>
<tr>
<td>GlideZipUtil</td>
<td>Packages.com.glide.util.ZipUtil</td>
</tr>
<tr>
<td>RhinoHelper</td>
<td>Packages.com.glide.script.RhinoHelper</td>
</tr>
<tr>
<td>SecurelyAccess</td>
<td>Packages.com.glide.sys.util.SecurelyAccess</td>
</tr>
<tr>
<td>ServiceAPI</td>
<td>Packages.com.glide.service_api.ServiceAPI</td>
</tr>
<tr>
<td>SncAddress32Bit</td>
<td>Packages.com.snc.commons.networks.Address32Bit</td>
</tr>
<tr>
<td>SncAliasApplier</td>
<td>Packages.com.snc.field_normalization.AliasApplier</td>
</tr>
<tr>
<td>SncAppFiles</td>
<td>Packages.com.snc.apps.api.AppFiles</td>
</tr>
<tr>
<td>SncApplicationFileListener</td>
<td>Packages.com.snc.apps.db.ApplicationFileListener</td>
</tr>
<tr>
<td>SncAppsUI</td>
<td>Packages.com.snc.apps.api.AppsUI</td>
</tr>
<tr>
<td>SncASensor</td>
<td>Packages.com.snc.discovery.sensor.ASensor</td>
</tr>
<tr>
<td>SncAuthentication</td>
<td>Packages.com.snc.authentication.digest.Authentication</td>
</tr>
<tr>
<td>SncBaselineCMDB</td>
<td>Packages.com.snc.cmdb.BaselineCMDB</td>
</tr>
<tr>
<td>SncBulkCopy</td>
<td>Packages.com.snc.ha.clone.BulkCopy</td>
</tr>
<tr>
<td>SncClassifiedProcess</td>
<td>Packages.com.snc.discovery.sensor.ClassifiedProcess</td>
</tr>
<tr>
<td>SncClassify</td>
<td>Packages.com.snc.discovery.sensor.snmp.Classify</td>
</tr>
<tr>
<td>SncCloneController</td>
<td>Packages.com.snc.ha.clone.CloneController</td>
</tr>
<tr>
<td>SncCloneInstance</td>
<td>Packages.com.snc.ha.clone.instance.Instance</td>
</tr>
<tr>
<td>SncCloneLogger</td>
<td>Packages.com.snc.ha.clone.CloneLogger</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>SncCloneTask</td>
<td>Packages.com.snc.ha.clone.CloneTask</td>
</tr>
<tr>
<td>SncCloneUtils</td>
<td>Packages.com.snc.ha.clone.CloneUtils</td>
</tr>
<tr>
<td>SncConfiguration</td>
<td>Packages.com.snc.field_normalization.db.Configuration</td>
</tr>
<tr>
<td>SncConfigurations</td>
<td>Packages.com.snc.field_normalization.Configurations</td>
</tr>
<tr>
<td>SncConnectionTest</td>
<td>Packages.com.snc.ha.connectivity.ConnectionTest</td>
</tr>
<tr>
<td>SncCloneFactoryHA</td>
<td>Packages.com.snc.db.clone.change.DBChangeManagerFactoryHA</td>
</tr>
<tr>
<td>SncDeviceHistory</td>
<td>Packages.com.snc.discovery.logging.DeviceHistory</td>
</tr>
<tr>
<td>SncDiscoveryCancel</td>
<td>Packages.com.snc.discovery.DiscoveryCancel</td>
</tr>
<tr>
<td>SncDiscoveryClassification</td>
<td>Packages.com.snc.discovery.DiscoveryClassification</td>
</tr>
<tr>
<td>SncDiscoveryLog</td>
<td>Packages.com.snc.discovery.logging.DiscoveryLog</td>
</tr>
<tr>
<td>SncDiscoveryRanges</td>
<td>Packages.com.snc.commons.networks.DiscoveryRanges</td>
</tr>
<tr>
<td>SncDiscoveryRangesDB</td>
<td>Packages.com.snc.discovery.DiscoveryRangesDB</td>
</tr>
<tr>
<td>SncDiscoveryReconciler</td>
<td>Packages.com.snc.discovery.database.DiscoveryReconciler</td>
</tr>
<tr>
<td>SncDiscoverySNMPClassification</td>
<td>Packages.com.snc.discovery.sensor.snmp.DiscoverySNMPClassification</td>
</tr>
<tr>
<td>SncDiscoveryUtils</td>
<td>Packages.com.snc.discovery.DiscoveryUtils</td>
</tr>
<tr>
<td>SncDropBackupTablesTask</td>
<td>Packages.com.snc.ha.clone.instance.DropBackupTablesTask</td>
</tr>
<tr>
<td>SncDropTablesTask</td>
<td>Packages.com.snc.ha.clone.DropTablesTask</td>
</tr>
<tr>
<td>SncEC2Properties</td>
<td>Packages.com.snc.ec2.EC2Properties</td>
</tr>
<tr>
<td>SncECMDBUtil</td>
<td>Packages.com.snc.cmdb.EMDBUtil</td>
</tr>
<tr>
<td>SncElrondClient</td>
<td>Packages.com.snc.customer_logon.ElrondClient</td>
</tr>
<tr>
<td>SncExtantDataJob</td>
<td>Packages.com.snc.field_normalization.db.ExtantDataJob</td>
</tr>
<tr>
<td>SncExtantDataJobState</td>
<td>Packages.com.snc.field_normalization.ExtantDataJobState</td>
</tr>
<tr>
<td>SncFailoverController</td>
<td>Packages.com.snc.da.gateway.replication.FailoverController</td>
</tr>
<tr>
<td>SncFileTree</td>
<td>Packages.com.snc.apps.file.FileTree</td>
</tr>
<tr>
<td>SncGatewayCache</td>
<td>Packages.com.snc.da.gateway.GatewayCache</td>
</tr>
<tr>
<td>SncGatewayClone</td>
<td>Packages.com.snc.da.gateway.clone.GatewayClone</td>
</tr>
<tr>
<td>SncGatewayConnectivity</td>
<td>Packages.com.snc.da.gateway.clone.GatewayConnectivity</td>
</tr>
<tr>
<td>SncGatewayPluginStartup</td>
<td>Packages.com.snc.da.gateway.replication.GatewayPluginStartup</td>
</tr>
<tr>
<td>SncGlideGateways</td>
<td>Packages.com.snc.da.gateway.GlideGateways</td>
</tr>
<tr>
<td>SncHAClone</td>
<td>Packages.com.snc.ha.clone.HAClone</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>SncHAConnectionTest</td>
<td>Packages.com.snc.ha.connectivity.HAConnectionTest</td>
</tr>
<tr>
<td>SncHADatabaseCheck</td>
<td>Packages.com.snc.ha.tablecheck.HADatabaseCheck</td>
</tr>
<tr>
<td>SncHAPairingUtils</td>
<td>Packages.com.snc.ha.HAPairingUtils</td>
</tr>
<tr>
<td>SncHAREplicationController</td>
<td>Packages.com.snc.ha.HAREplicationController</td>
</tr>
<tr>
<td>SncHAREplicationQueueSnapshotBuilder</td>
<td>Packages.com.snc.ha.tablecheck.HAREplicationQueueSnapshotBuilder</td>
</tr>
<tr>
<td>SncHATableCheck</td>
<td>Packages.com.snc.ha.tablecheck.HATableCheck</td>
</tr>
<tr>
<td>SncHATableCheckThread</td>
<td>Packages.com.snc.ha.tablecheck.HATableCheckThread</td>
</tr>
<tr>
<td>SncHATableQuickCheck</td>
<td>Packages.com.snc.ha.tablecheck.HATableQuickCheck</td>
</tr>
<tr>
<td>SncHATableRepair</td>
<td>Packages.com.snc.ha.tablecheck.HATableRepair</td>
</tr>
<tr>
<td>SncHAUtils</td>
<td>Packages.com.snc.ha.HAUtils</td>
</tr>
<tr>
<td>SncHostname</td>
<td>Packages.com.snc.discovery.utils.Hostname</td>
</tr>
<tr>
<td>SncInstanceClone</td>
<td>Packages.com.snc.clone.instance.InstanceClone</td>
</tr>
<tr>
<td>SncInstanceConnectionTest</td>
<td>Packages.com.snc.ha.connectivity.InstanceConnectionTest</td>
</tr>
<tr>
<td>SncInstanceRollback</td>
<td>Packages.com.snc.clone.instance.InstanceRollback</td>
</tr>
<tr>
<td>SncIPAddressV4</td>
<td>Packages.com.snc.commons.networks.IPAddressV4</td>
</tr>
<tr>
<td>SncIPAddressV6</td>
<td>Packages.com.snc.commons.networks.IPAddressV6</td>
</tr>
<tr>
<td>SncIPAuthenticator</td>
<td>Packages.com.snc.ipauthenticator.IPAuthenticator</td>
</tr>
<tr>
<td>SncIPIterator</td>
<td>Packages.com.snc.commons.networks.IPIterator</td>
</tr>
<tr>
<td>SncIPLIST</td>
<td>Packages.com.snc.commons.networks.IPLIST</td>
</tr>
<tr>
<td>SncIPMetaCollection</td>
<td>Packages.com.snc.commons.networks.IPMetaCollection</td>
</tr>
<tr>
<td>SncIPNetmaskV4</td>
<td>Packages.com.snc.commons.networks.IPNetmaskV4</td>
</tr>
<tr>
<td>SncIPNetworkV4</td>
<td>Packages.com.snc.commons.networks.IPNetworkV4</td>
</tr>
<tr>
<td>SncIPRangeV4</td>
<td>Packages.com.snc.commons.networks.IPRangeV4</td>
</tr>
<tr>
<td>SncJRobinGraphDef</td>
<td>Packages.com.snc.jrobin._JRooBinGraphDef</td>
</tr>
<tr>
<td>SncLayer7Connections</td>
<td>Packages.com.snc.discovery.Layer7Connections</td>
</tr>
<tr>
<td>SncMACAddress</td>
<td>Packages.com.snc.commons.networks.MACAddress</td>
</tr>
<tr>
<td>SncMACAddressMfr</td>
<td>Packages.com.snc.commons.networks.MACAddressMfr</td>
</tr>
<tr>
<td>SncMakeAndModel</td>
<td>Packages.com.snc.cmdb.MakeAndModel</td>
</tr>
<tr>
<td>SncMIDConfigParameter</td>
<td>Packages.com.snc.commons.MIDConfigParameter</td>
</tr>
<tr>
<td>SncMIDServerRangesDB</td>
<td>Packages.com.snc.discovery.MIDServerRangesDB</td>
</tr>
<tr>
<td>SncNormalCoalescer</td>
<td>Packages.com.snc.field_normalization.NormalCoalescer</td>
</tr>
<tr>
<td>SncNormalizer</td>
<td>Packages.com.snc.field_normalization.Normalizer</td>
</tr>
<tr>
<td>SncNormalValueChanger</td>
<td>Packages.com.snc.field_normalization.NormalValueChanger</td>
</tr>
<tr>
<td>SncNotifySNC</td>
<td>Packages.com.snc.system.NotifySNC</td>
</tr>
<tr>
<td>GlideScriptable Class</td>
<td>Packages Call</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SncOnCallRotation</td>
<td>Packages.com.snc.on_call_rotation.OnCallRotation</td>
</tr>
<tr>
<td>SncPendingValueCollector</td>
<td>Packages.com.snc.field_normalization.PendingValueCollector</td>
</tr>
<tr>
<td>SncPreferences</td>
<td>Packages.com.snc.field_normalization.Preferences</td>
</tr>
<tr>
<td>SncPrintServerHelper</td>
<td>Packages.com.snc.discovery.database.PrintServerHelper</td>
</tr>
<tr>
<td>SncProbe</td>
<td>Packages.com.snc.discovery.Probe</td>
</tr>
<tr>
<td>SncProbeRunTime</td>
<td>Packages.com.snc.discovery.perfmon.ProbeRunTime</td>
</tr>
<tr>
<td>SncRBSensorProcessor</td>
<td>Packages.com.snc.discovery_automation.RBSensorProcessor</td>
</tr>
<tr>
<td>SncReadTest</td>
<td>Packages.com.snc.ha.ReadTest</td>
</tr>
<tr>
<td>SncReclassifyCI</td>
<td>Packages.com.snc.cmdb.ReclassifyCI</td>
</tr>
<tr>
<td>SncRelationships</td>
<td>Packages.com.snc.cmdb.Relationships</td>
</tr>
<tr>
<td>SncReplicationAdvisor</td>
<td>Packages.com.snc.db.replicate.ReplicationAdvisor</td>
</tr>
<tr>
<td>SncReplicationEngine</td>
<td>Packages.com.snc.db.replicate.ReplicationEngine</td>
</tr>
<tr>
<td>SncReplicationQueue</td>
<td>Packages.com.snc.db.replicate.ReplicationQueue</td>
</tr>
<tr>
<td>SncRequestCredentials</td>
<td>Packages.com.snc.customer_logon.RequestCredentials</td>
</tr>
<tr>
<td>SncRuleApplier</td>
<td>Packages.com.snc.field_normalization.RuleApplier</td>
</tr>
<tr>
<td>SncRuleToPending</td>
<td>Packages.com.snc.field_normalization.RuleToPending</td>
</tr>
<tr>
<td>SncSAMCounter</td>
<td>Packages.com.snc.software_asset_management.SAMCounter</td>
</tr>
<tr>
<td>SncScheduleDropBackupTablesTask</td>
<td>Packages.com.snc.ha.clone.instance.ScheduleDropBackupTablesTask</td>
</tr>
<tr>
<td>SncScrapeIANAEnterpriseNumbers</td>
<td>Packages.com.snc.discovery.database.ScrapeIANAEnterpriseNumbers</td>
</tr>
<tr>
<td>SncScrapeIEEENICCodes</td>
<td>Packages.com.snc.discovery.database.ScrapeIEEENICCodes</td>
</tr>
<tr>
<td>SncSendNotificationTask</td>
<td>Packages.com.snc.ha.clone.instance.SendNotificationTask</td>
</tr>
<tr>
<td>SncSensorProcessor</td>
<td>Packages.com.snc.discovery.SensorProcessor</td>
</tr>
<tr>
<td>SncSerialNumber</td>
<td>Packages.com.snc.discovery.SerialNumber</td>
</tr>
<tr>
<td>SncSerialNumberList</td>
<td>Packages.com.snc.discovery.SerialNumberList</td>
</tr>
<tr>
<td>SncSessionMate</td>
<td>Packages.com.snc.discovery.SessionMate</td>
</tr>
<tr>
<td>SncSimmerControl</td>
<td>Packages.com.snc.ha.clone.instance.SimmerControl</td>
</tr>
<tr>
<td>SncTableEditor</td>
<td>Packages.com.snc.apps.api.TableEditor</td>
</tr>
<tr>
<td>SncTableRotation</td>
<td>Packages.com.snc.db.replicate.TableRotation</td>
</tr>
<tr>
<td>SncTableRotationExtension</td>
<td>Packages.com.snc.db.replicate.TableRotationExtension</td>
</tr>
<tr>
<td>SncTableRotationExtensions</td>
<td>Packages.com.snc.db.replicate.TableRotationExtensions</td>
</tr>
<tr>
<td>SncTableRotationWatcher</td>
<td>Packages.com.snc.db.replicate.TableRotationWatcher</td>
</tr>
<tr>
<td>SncTransformApplier</td>
<td>Packages.com.snc.field_normalization.TransformApplier</td>
</tr>
<tr>
<td>SncTreeBuilder</td>
<td>Packages.com.snc.apps.tree.TreeBuilder</td>
</tr>
</tbody>
</table>
### GlideScriptable Class Packages Call

<table>
<thead>
<tr>
<th>GlideScriptable Class</th>
<th>Packages Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>SncTriggerSynchronizer</td>
<td>Packages.com.snc.automation.TriggerSynchronizer</td>
</tr>
<tr>
<td>SncValue</td>
<td>Packages.com.snc.field_normalization.db.Value</td>
</tr>
<tr>
<td>TestExtension</td>
<td>Packages.com.glide.junit.misc.TestExtension</td>
</tr>
<tr>
<td>UINotification</td>
<td>Packages.com.glide.ui.UINotification</td>
</tr>
</tbody>
</table>

### Packages Call Removal Tool error types

Possible error types generated by the Packages Call Removal Tool.

**Error types**

<table>
<thead>
<tr>
<th>Error message</th>
<th>Description</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Name Could Not Be Parsed</td>
<td>A line of script has what appears to be a Packages call (for example, the line contains Packages.com.glide) but on examination there is insufficient text to parse out a class name that corresponds to a scriptable object name.</td>
<td>This error type generally does not present a problem, but is nevertheless flagged as an error in case the script references a valid class that the parser, for whatever reason, cannot parse.</td>
</tr>
<tr>
<td>Class Does Not Exist</td>
<td>A line of script has an identifiable Packages call to a Java class that does not exist. This may be because it was incorrectly typed, or because the Java class no longer exists. Either way, the line of script is currently not doing anything and is potentially generating errors whenever it is run.</td>
<td>This error type usually identifies a script that is not doing what its author intended, if it ever did. The original script should be revisited, and the invalid Packages call removed.</td>
</tr>
<tr>
<td>Class Is Not a Scriptable Class</td>
<td>A line of script has an identifiable Packages call to an existing ServiceNow Java class that is not currently marked as scriptable. The call cannot be replaced, because there is currently no corresponding scriptable name.</td>
<td>To convert this script, either rewrite the script so that it does not use the Java class or wait until ServiceNow, Inc., provides a scriptable name for the class.</td>
</tr>
<tr>
<td>Class Does Not Have a Scriptable Name</td>
<td>A line of script has an identifiable Packages call to an existing ServiceNow Java class that does not currently have a scriptable name.</td>
<td>This error is less likely to occur than a Class Is Not a Scriptable Class error; however, as with that error, to convert this script, either rewrite the script so that it does not use the Java class or wait until ServiceNow, Inc., provides a scriptable name for the class.</td>
</tr>
</tbody>
</table>
### Error message

<table>
<thead>
<tr>
<th>Variable Name Being Assigned Conflicts With a Scriptable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A variable name has the same name as a scriptable name. For example, <code>var GlideSession = Packages.com.glide.sys.GlideSession.get();</code> would generate this error because the variable name <code>GlideSession</code> is the same as the scriptable name for the GlideSession Java class.</td>
</tr>
<tr>
<td>Before this script can be converted, the variable name must be changed wherever it occurs in the script. For this example, the line would be changed to <code>var gsession = Packages.com.glide.sys.GlideSession.get();</code> to remove the conflict. Also, be sure to change the variable name elsewhere in the script wherever it is used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Internal Error - Unable To Find String To Be Replaced</th>
</tr>
</thead>
<tbody>
<tr>
<td>The tool is unable to find the script text it is currently evaluating for Packages call replacement.</td>
</tr>
<tr>
<td>This error type is very unlikely. If it does occur, examine your code and correct any errors. If the error message still occurs, open an incident with Technical Support.</td>
</tr>
</tbody>
</table>

### Web services

HTTP-based web services allow diverse applications to talk to each other. ServiceNow supports both inbound (provider) and outbound (consumer) web services.

#### Direct inbound web services

Inbound web services allow you to access and modify ServiceNow data using a client application.

- REST API
- Scripted REST APIs
- SOAP web service
- CSV Web service
- Excel web service
- JSONv2 Web Service
- PDF web service
- RSS feed generator
- XML web service

#### Other inbound web services

- ODBC driver
- Scripted SOAP web services
Outbound web services

Outbound web services allow you to send SOAP and REST messages to external web service providers.

- Outbound SOAP web service
- Outbound REST web service

REST API

REST (REpresentational State Transfer) is a simple stateless architecture that generally runs over HTTP.

The REST style emphasizes that interactions between clients and services are enhanced by having a limited number of operations. Flexibility is provided by assigning resources their own unique universal resource indicators (URIs). Because each operation (GET, POST, PUT, and DELETE) has a specific meaning, REST avoids ambiguity.

The REST API is active by default in all instances.

RESTful web services offer administrators several advantages, including:

- Support for different HTTP methods to perform different actions
- Detailed response codes and header information
- Pagination support for large data sets
- Streaming data on GET requests

The Automated Test Framework supports Inbound REST test steps. You can create automated tests for custom Inbound REST APIs that you create. Creating tests for your custom REST APIs simplifies upgrade testing, and makes it possible to verify modifications to a REST API are backward compatible. See Administering REST test step configurations and ATF REST test step configurations.

Developer training

On the ServiceNow® Developer Site, you can get training for Inbound REST integrations and Outbound REST integrations.

Getting started with the REST API

To become familiar with ServiceNow REST API support, review these example scenarios and test the REST API functionality.

In this tutorial you will use the REST API Explorer to test the REST API. The REST API Explorer allows you to discover ServiceNow REST APIs, quickly construct and execute requests, and view responses from ServiceNow REST APIs within your browser.

Before beginning, ensure your user account has the rest_api_explorer and web_service_admin roles. These roles are required to complete the example procedures.

Access the REST API Explorer

View available REST API resources using the REST API Explorer.

Role required: web_service_admin, rest_api_explorer, or admin

You can browse available APIs, API versions, and methods for each API.

Navigate to System Web Services > REST API Explorer.
### Table API

Allows you to perform create, read, update and delete (CRUD) operations on existing tables.

#### Prepare request

**Request Headers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Format</td>
<td>JSON</td>
<td>Format of the REST request body</td>
</tr>
<tr>
<td>Response Format</td>
<td>JSON</td>
<td>Format of the REST response body</td>
</tr>
<tr>
<td>Authorization</td>
<td>Send as me</td>
<td>Send the request as the current user or with another user's credentials</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Path Parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>tableName</td>
<td>incident</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query Parameters</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_query</td>
<td></td>
<td>An encoded query string used to filter the results</td>
</tr>
<tr>
<td>sysparm_display_value</td>
<td></td>
<td>Return the display value (true), actual value (false), or both (all) for reference fields (default: false)</td>
</tr>
<tr>
<td>sysparm_exclude_reference_link</td>
<td></td>
<td>True to exclude Table API links for reference fields (default: false)</td>
</tr>
</tbody>
</table>
**Retrieve existing incidents**

Use a GET request to view existing incident records.

Role required: web_service_admin, rest_api_explorer, or admin

Use the REST API Explorer to send the following request:

```plaintext
GET https://instance.service-now.com/api/now/v1/table/incident
```

1. In the top-left of the REST API Explorer, select **Table API** and **v1 version**.
2. Click **Retrieve records from a table (GET)**. For information, see Table API - GET table
3. In the **Path Parameters** section, select the incident table.
4. Scroll to the bottom of the page and click **Send**.

The response includes incident records from the instance. The REST API Explorer limits queries to 10 records at a time. Only the first 10 incident records appear. The response also includes a Link header that provides the URL to query the next 10 incident records.
Response

Status code 200 OK

Headers

Content-Encoding gzip
Content-Type application/json
Date Tue, 08 Oct 2015 08:24:46 GMT
Server ServiceNow
Transfer-Encoding chunked
X-Total-Count 54

Response Body

```json
{
  "result": [
    {
      "upon_approval": "",
      "location": {
        "Link": "https://service-now.com/api/now/table/cmn_location/1083361cc61127581b682158cabf646",
        "value": "1083361cc61127581b682158cabf646"
      },
      "expected_start": "",
      "reopen_count": "",
      "close_notes": "",
      "impact": "1",
      "urgency": "1",
      "correlation_id": "",
      "sys_tags": "",
      "sys_domain": {
        "Link": "https://service-now.com/api/now/table/sys_user_group/global",
```
```
Create an incident record

Use a POST request to create a new record.

Role required: web_service_admin, rest_api_explorer, or admin

Use the REST API Explorer to send the following request:

POST https://instance.service-now.com/api/now/v1/table/incident

1. In the top-left of the REST API Explorer, click Create a record (POST).
2. In the Path Parameters section, select the incident table.
3. In the Request Body section, click Add a field.
4. Select a field and specify a value for that field.
5. Optional: Click Add another field and specify an additional field to assign a value to.

   The request body updates automatically based on your entries, such as
   "short_description":"Test incident creation through REST",
   "comments":"These are my comments"

6. After constructing the request, click Send.

   The response includes a Location header that specifies where the incident was created and
   how to retrieve the incident. Record this header to use in the next part of this guide.
## Response

<table>
<thead>
<tr>
<th>Status code</th>
<th>201 Created</th>
</tr>
</thead>
</table>

### Headers

<table>
<thead>
<tr>
<th>Content-Encoding</th>
<th>gzip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>application/json</td>
</tr>
<tr>
<td>Date</td>
<td>Tue, 08 Sep 2015 12:11:03 GMT</td>
</tr>
<tr>
<td>Location</td>
<td>https://&lt;instance&gt;.servicenow.com/api/noc/v1/table/incident/ba8f71b22b1a0200b6706b7219da154f</td>
</tr>
<tr>
<td>Server</td>
<td>ServiceNow</td>
</tr>
<tr>
<td>Transfer-Encoding</td>
<td>chunked</td>
</tr>
</tbody>
</table>

### Response Body

```
"short_description": "Test incident creation through REST",
"order": "",
"sys_updated_by": "admin1",
"resolved_by": "",
"notify": "1",
"upon_reject": "cancel",
"approval_history": "",
"problem_id": "",
"work_notes": "",
"calendar_duration": "",
"close_code": "",
"sys_id": "ba8f71b22b1a0200b6706b7219da154f",
```

---

### Read the inserted incident

Use the Location header from the POST response to run a GET request.

Role required: web_service_admin, rest_api_explorer, or admin
Use the REST API Explorer to send the following request:

GET https://instance.service-now.com/api/now/v1/table/incident/{sys_id}

1. In the top-left of the REST API Explorer, click **Retrieve a record (GET)**.
2. In the **Path Parameters** section, select the incident table.
3. In the **sys_id** field, enter the sys_id of the record you created.
   The record sys_id appears as a 32-character string at the end of the POST response Location header.
4. Click **Send**.
   The response body contains a text representation of the record. You can control the format of the response, such as JSON or XML, using the **Response Format** field.
Update the incident

Update the incident record using either a PUT or PATCH function.

Role required: web_service_admin, rest_api_explorer, or admin

Use the REST API Explorer to send the following request:
PUT https://instance.service-now.com/api/now/v1/table/incident/{sys_id}?sysparm_exclude_ref_link=true

1. In the top-left of the REST API Explorer, click Modify a record (PUT) or Update a record (PATCH).
2. In the Path Parameters section, select the incident table.
3. In the sys_id field, enter the sys_id of the record you created.
4. In the Request Body section, click Add a field.
5. Select the Short description field and specify a new value.
6. Click Send.
7. Verify that the response contains the updated short_description value.

Response

<table>
<thead>
<tr>
<th>Status code</th>
<th>200 OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headers</td>
<td></td>
</tr>
<tr>
<td>Content-Encoding</td>
<td>gzip</td>
</tr>
<tr>
<td>Content-Type</td>
<td>application/json</td>
</tr>
<tr>
<td>Date</td>
<td>Tue, 08 Sep 2015 12:17:08 GMT</td>
</tr>
<tr>
<td>Server</td>
<td>ServiceNow</td>
</tr>
<tr>
<td>Transfer-Encoding</td>
<td>chunked</td>
</tr>
</tbody>
</table>

Response Body

```
"short_description": "This is a different short description",
"order": ",",
"sys_updated_by": "admin1",
"resolved_by": ",",
"notify": "1",
"upon_reject": "cancel",
"approval_history": ",",
"problem_id": ",",
"work_notes": ",",
"calendar_duration": ",",
"close_code": ",",
"sys_id": "ba8f71b22b1a0200b6706b7219da154f",
```
Delete the incident

Delete the incident using a DELETE request.

Role required: web_service_admin, rest_api_explorer, or admin

Use the REST API Explorer to send the following request:

DELETE https://instance.service-now.com/api/now/v1/table/incident/{sys_id}

1. In the top-left of the REST API Explorer, click **Delete a record (DELETE)**.
2. In the **Path Parameters** section, select the incident table.
3. In the **sys_id** field, enter the sys_id of the record you created.
4. Click **Send**.
5. Verify that the response status code is 204.

**REST API Explorer**

The REST API Explorer uses information from your instance to provide you with a list of endpoints, methods, and variables that you can use to build and send a REST request.

After you build the request, the REST API Explorer provides code samples in multiple programming languages that you can use to send the request, and detailed request and response information.

Access the REST API Explorer by navigating in your instance to **System Web Services > REST API Explorer**. You must have the rest_api_explorer role to access the REST API Explorer.

**Warning:** The REST API Explorer interacts with data on the current instance. Use caution when working with requests that create, edit, or delete data on a production instance.

**Limitations**

In the Request Body in REST API Explorer, you can't build a complex array of objects on the Builder tab. You can use only one object on the Builder tab. You must use the Raw tab to build a complex array of objects.

**REST API Explorer elements**

The REST API Explorer displays multiple elements that allow you to construct and test REST requests. The Namespace, API, method, Request Headers and Path Parameters sections appear for all requests. Other sections may appear depending on the selected API and method.

**Namespace, API, and method**

The choice lists at the top of the REST API Explorer allow you to select an API, such as the Table API, Aggregate API, or Import Set API, and an API version. Available APIs depend on the selected Namespace. REST APIs provided by ServiceNow use the now namespace. Scripted REST Services may use a different namespace.

Links that appear below these choice lists display available methods for the selected API and version, such as the GET method for the Table API. Each method includes a brief description of the action that method performs. The REST API Explorer displays a sample endpoint and certain Request Elements based on the selected API and method.
Request headers

In the Request Headers section, specify the HTTP request and response headers. To add a custom header, click Add custom header and specify the header name and value.

Path parameters

In the Path Parameters section, select the table to interact with. For the Import Set API, import staging tables are available. For the Table API and Aggregate API, all tables that allow web service interaction are available. For requests that interact with a single existing record, such as those that modify a record, the sys_id field allows you to specify the record.

Query parameters

In the Query Parameters section, define parameters to modify the request, such as the encoded query used to filter results when performing a Table API GET request.

The parameters that appear depend on the selected request method. To add custom query parameters, click Add query parameter and specify the parameter name and value. Refer to the documentation for each REST API method for detailed information about the available parameters.

Request body

In the Request Body section, define the body content of the request. This section appears for methods that insert or update data, such as PUT. The Request Format that you select in the Request Headers section determines the format you must use in the request body, such as JSON or XML.

You can build the request body by selecting fields, by manually writing the request body, or by uploading a binary file.

Use the Builder tab to build the request body using instance fields. Click Add a field, then select a field and specify a value. Available fields depend on the table you are adding data to.

Use the Raw tab to manually enter a request body.

Use the Binary tab to upload a binary file. This tab appears only for APIs that support binary file data, such as the Attachment API.

Code samples

You can obtain a code sample in various languages that you can use to send the request. Links for code samples appear at the bottom of the form. Click a link to view or copy the code sample for that language. The code samples change based on data you enter on the form. Code samples are available in these languages:

- ServiceNow Script
- cURL
- Python
- Ruby
- JavaScript
- Perl
• Powershell

Request and response details

After sending a request, the Request and Response sections appear providing detailed information including the full request URI, headers for both the request and response, the HTTP status code of the response, and the response body. Click Clear Response to hide this information.

When testing a request that returns a binary response, such as a file attachment, the REST API Explorer uses the browser default functionality to handle the file, such as by prompting you to download the file.

Explore the REST API for a table

You can explore the REST API for a table directly from that table. Explore the API using the REST API Explorer to quickly construct and test REST requests for that table.

Role required: itil, personalize_dictionary, and rest_api_explorer

1. Navigate to any form or list.
2. In a form, right-click the form header and select Configure > Table.
3. In a list, perform the appropriate action for the list version.
   • For List v2, right-click any column heading and select Configure > Table.
   • For List v3, open the list title menu and select Configure, and then click Table.
4. Click the Explore REST API related link.
   The REST API Explorer opens, displaying the Table API for the selected table. If the table does not allow web service interaction, a warning appears instead.

Use the REST API Explorer to construct and test REST requests for the table.

Example REST client applications

Several example REST client applications and source code are available to demonstrate integrations using REST web services.

The example REST client applications demonstrate how to use the ServiceNow REST API with an external application, such as a NodeJS or iOS application.

Important: These applications are provided only as a demonstration of the REST functionality and are not officially supported.

The examples are open source and available to the community. You can use these example applications to help familiarize yourself with the REST functionality, or use them as a starting point to create your own REST client applications.

Users with the rest_api_explorer role can access the list of available client applications by navigating to System Web Services > REST > Example Client Apps.

When viewing the list of applications, click an application to view the source code and additional documentation hosted on GitHub.
REST API version

REST API URLs may specify a version number, such as `/api/now/v1/table/{tableName}` or no version number, such as `/api/now/table/{tableName}`.

URIs with a version number provide a consistent interface that is guaranteed to maintain backwards compatibility in future ServiceNow releases. For example, if the default response format for a particular HTTP method changes in a release, versioned URIs continue to use the previous format.

By specifying a version number in your URIs, you can ensure that future updates to the REST API do not negatively impact your integration. URIs that do not specify a version number use the most recent REST API behavior available with your instance version, which may change when you upgrade.

REST API security

By default, the REST API uses basic authentication or OAuth to enforce access controls to web resources.

Access control lists (ACLs) defined for tables are enforced to restrict access to data.

The user ID that is used for authentication is subject to access control in the same way as an interactive user. Each request requires the proper authentication information. Ensure that each request includes an Authorization header with the credentials you want to use. There is no support for inbound mutual authentication.

To allow access to tables without any authentication and authorization, add the table name to the Public Pages (sys_public) table with a status of Active. ACLs defined on tables are still enforced, and it is the customer's responsibility to deactivate ACLs on tables.

REST supports cookies for binding to the existing session.

REST API ACLs

ServiceNow-defined ACLs are defined for the following REST APIs:

- Table API
- Aggregate API
- Import Set API
- Attachment API

The ACLs have the same name as the service. For example, the ACL defined for the REST Table API is named Table API.

Note: The names of these ACLs should never be changed or modified.

These ACLs are deactivated by default, but can be activated on a per API basis. If the REST API ACL is activated for a platform REST API, a user must have the snc_platform_rest_api_access role to make a request to that REST API.

Note: The snc_platform_rest_api_access role replaces the deprecated rest_service role.

REST API roles

Certain roles are required for inbound REST operations.
### Inbound REST API roles

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
</table>
| snc_platform_rest_api_access | Controls the ability to access the following services:  
| | · Table API  
| | · Aggregate API  
| | · Import Set API  
| | · Attachment API |

**Important:** To perform database operations using REST APIs, you must have any roles needed to access the target tables.

### Attachment API role requirements

The user record used to authenticate Attachment API requests must have certain roles.

To create attachments, the user record used to authenticate the HTTP request with ServiceNow must have any roles required to create Attachment (sys_attachment) records, and also have any roles required to read and write records on the target table, such as the itil role to add attachments to incident records.

By default there is no single role allowing a user to add attachments. You can create a role to explicitly allow adding attachments, then assign this role to the user account being used to make the request.

### REST API table access

All tables, including base system tables, global tables, and scoped tables are accessible via web services by default.

You must fulfill any other web service security requirements, such as basic authentication and ACLs to access tables via web services.

You can control direct web service access to tables using the **Allow access to this table via web services** check box on the table application access settings. This check box must be selected to allow web service interaction with the table.

**Note:** The application access fields controlling CRUD operations, such as **Can read** or **Can create** do not apply to web service requests.

### REST API CORS support

The REST API supports cross-origin resource sharing (CORS) security.

CORS support allows you to define which domains can access each REST API. By defining a CORS rule, you can whitelist a domain to allow cross-origin requests from that domain. Cross-origin requests cannot be made from domains without a CORS rule. If no CORS rule is configured, an endpoint/resource fails if the request is not coming from the same origin.

**Note:** CORS support applies only to REST APIs, including scripted REST web services. Other web service APIs, such as the SOAP API, do not support CORS.
You can configure CORS to allow access to only certain APIs, HTTP methods, and headers from other domains. For example, you can limit requests to the Table API from a specific domain to allow only GET operations.

To view the CORS rules defined on your instance, navigate to **System Web Services > CORS Rules**.

You can disable CORS support for an instance by setting the property `glide.rest.cors.enabled` to `false`. When `false`, no CORS evaluation is performed on incoming REST requests. This property is `true` by default.

**Note:** You must add this property to the System Property (sys_properties) table.

### Define a CORS rule

You can define a CORS rule to control which domains can access specific REST API endpoints.

**Role required:** web_service_admin

1. Navigate to **System Web Services > CORS Rules**.
2. Click **New**.
3. Populate the form.

<table>
<thead>
<tr>
<th>Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST API</td>
<td>Select the REST API this CORS rule applies to, such as the Table API.</td>
</tr>
<tr>
<td>Domain</td>
<td>Enter the domain that this CORS rule applies to. This CORS rule is evaluated against requests from the specified domain. You can specify a domain pattern or an IP address. When using a domain pattern you can specify a single wildcard to match incoming origin headers.</td>
</tr>
<tr>
<td>HTTP Methods</td>
<td>Select the HTTP methods allowed. Only the selected methods can be called from the specified domain.</td>
</tr>
<tr>
<td>HTTP Headers</td>
<td>Enter a comma-separated list of HTTP headers to send in the response. Specified headers are added to the Access-Control-Expose-Headers header.</td>
</tr>
<tr>
<td>Max age</td>
<td>Enter the number of seconds to cache the client session. After an initial CORS request, further requests from the same client within the specified time do not require a preflight message. If you do not specify a value, the default value of 0 indicates that all requests require a preflight message.</td>
</tr>
</tbody>
</table>

4. Click **Submit**.

### CORS domain requirements

When you define a CORS rule, the value you enter in the **Domain** field must meet certain requirements. Each CORS rule supports a single wildcard to match incoming Origin headers.
Requirements

The value you enter in the **Domain** field must meet the following requirements.

- Begins with **HTTP://** or **HTTPS://**.
- Is a domain pattern or IP address.
- Ends with alphanumeric characters preceded by a period, such as `.com`.
- Includes at most a single wildcard character immediately following the scheme and hierarchical portion of the domain pattern.

Wildcard

You can use a single wildcard character (*) in the domain pattern. Use this wildcard immediately following the scheme and hierarchical portion of the domain pattern, such as `http://*.domain.com` to include all subdomains. The wildcard must immediately follow the scheme and hierarchical portion of the domain pattern. If you use an IP address instead of domain pattern, you must enter the full IP address without a wildcard.

**Note:** You cannot use multiple wildcards, or specify a wildcard without a domain pattern. Values such as `*` or `*.` are not supported.

Domain matching

When evaluating the Origin header in a request, ServiceNow prioritizes rules that match the domain pattern exactly. If no exact match is found, the next closest match is used.

For example, if there are rules for the domain patterns `http://*.blog.mysite.com` and `http://*.mysite.com`, a request from `http://alice.blog.mysite.com` will match the `http://*.blog.mysite.com` pattern.

Examples of valid and invalid domains

<table>
<thead>
<tr>
<th>Valid domain</th>
<th>Invalid domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>http://*.ms.net</td>
<td>https://*com</td>
</tr>
<tr>
<td>https://*.ms.com</td>
<td>http://*..com</td>
</tr>
<tr>
<td><a href="http://192.168.1.1">http://192.168.1.1</a></td>
<td>http://*.168.1.126</td>
</tr>
<tr>
<td>http://*.service-now.com</td>
<td><a href="http://blog.*.service-now.com">http://blog.*.service-now.com</a></td>
</tr>
<tr>
<td>http://*.com</td>
<td>http://*com</td>
</tr>
</tbody>
</table>

Enable OAuth with inbound REST

Using OAuth, you can pass a user ID and password once, and then use a token for subsequent REST requests instead of submitting credentials with each request.
OAuth can improve system security by reducing the number of times you submit user credentials. You can use OAuth to authenticate REST requests.

This video demonstrates how to authenticate to REST APIs using OAuth.

1. Activate the OAuth 2.0 plugin.
2. Set the system property `com.snc.platform.security.oauth.is.active` to true.
3. Navigate to System OAuth > Application Registry.
4. Click New and then click Create an OAuth API endpoint for external clients.
5. Record the client_id and client_secret values from the previous step to use when requesting an access token.
6. To get an access token, use your REST client, such as cURL or Postman, to send a request to the OAuth endpoint (oauth_token.do).
   
   Format the request as a URL-encoded HTTP POST body and include the required parameters.
7. Record the access token and refresh token from the response.
8. Submit the access token with subsequent REST requests.

REST OAuth example

This example shows how to authenticate an inbound REST request using OAuth.

In this example, the OAuth token has a client_id of `a329c4515612210071a5e0c298ee2be8` and a client_secret of `password22`.

Getting an access token

```
curl -d "grant_type=password&client_id=a329c4515612210071a5e0c298ee2be8&client_secret=password22&username=RESTUser&password=RESTUserPassword" https://<instance>.service-now.com/oauth_token.do
```

Sample token response

```
{
  "scope": "useraccount",
  "token_type": "Bearer",
  "expires_in": 1799,
  "refresh_token": "j2PdkEvRrW vtMj8spldJN1S0uWM4D7QV9m3cQXDVo5Qa_GVvmdR6NOp70M03EHJnd6n2pWocFer1NcJz4zwvW",
  "access_token": "2wRlsRCT2SYjCCJP91kwo2EFzj5qg4O3I3aC09e0-0hz6Ib3YK7If-LMiNurNuqfqbkl4AfkyC92KYHUCcbPQ"
}
```

REST request with OAuth token

```
curl -H "Accept:application/json" -H "Authorization:Bearer 2wRlsRCT2SYjCCJP91kwo2EFzj5qg4O3I3aC09e0-0hz6Ib3YK7If-LMiNurNuqfqbkl4AfkyC92KYHUCcbPQ" "https://<instance>.service-now.com/api/now/table/incident"
```
Multi-factor authentication for inbound REST

You can submit a multi-factor authentication token with user credentials to authenticate an MFA-enabled user account using REST.

When multi-factor authentication is enabled for a user account, you must submit an MFA token with basic auth credentials when making REST requests as that user.

To send an MFA token with a REST request, append the token to the end of the user's password in the basic auth username:password string, such as joe.employee:password62161147. Encode the full string including the MFA token using base64 encoding, then send the encoded string in the Authorization header.

Multi-factor authentication REST responses

The response to an MFA authentication request varies depending on the validity of the provided credentials and MFA token.

<table>
<thead>
<tr>
<th>Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic auth credentials and MFA token are valid</td>
<td>The user is authenticated and the session created. The request is processed normally.</td>
</tr>
<tr>
<td>Basic auth credentials are valid but MFA token is invalid or missing</td>
<td>The response returns error 401. The response includes the X-MFA_TOKEN header with the value &quot;invalid&quot;.</td>
</tr>
<tr>
<td>Basic auth credentials are invalid</td>
<td>The response returns error 401. The X-MFA_TOKEN header is not included in the response.</td>
</tr>
</tbody>
</table>

Supported REST API headers

The REST API provides support for various headers. Some headers are mandatory for specific APIs and HTTP methods.

Data format headers

The Accept and Content-Type request headers are required for proper data formatting for requests that contain a request body or response body. These request headers support the following values:

- Accept: application/json, application/xml
- Content-Type: application/json, application/xml

**Note:** The Attachment API accepts all Content-Type values (*/"). Specify the file content type when uploading an attachment. The content type is stored with file metadata allowing other tools to correctly identify and parse the file.

POST, PUT, PATCH, and DELETE operations require you to provide both headers. GET operations require only the Accept header. Failing to provide the required headers results in a 400 Bad Request error.
Other headers

All requests may contain an authentication header that specifies the user credentials to authenticate with.

You can override the HTTP method, such as GET or POST, by setting the X-http-method-override header.

REST API HTTP response codes

REST Messages sent to an instance return a specific HTTP response code.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Message</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
<td>Success with response body.</td>
</tr>
<tr>
<td>201</td>
<td>Created</td>
<td>Success with response body.</td>
</tr>
<tr>
<td>204</td>
<td>Success</td>
<td>Success with no response body.</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>The request URI does not match the APIs in the system, or the operation failed for unknown reasons. Invalid headers can also cause this error.</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>The user is not authorized to use the API.</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>The requested operation is not permitted for the user. This error can also be caused by ACL failures, or business rule or data policy constraints.</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
<td>The requested resource was not found. This can be caused by an ACL constraint or if the resource does not exist.</td>
</tr>
<tr>
<td>405</td>
<td>Method not allowed</td>
<td>The HTTP action is not allowed for the requested REST API, or it is not supported by any API.</td>
</tr>
<tr>
<td>406</td>
<td>Not acceptable</td>
<td>The endpoint does not support the response format specified in the request Accept header.</td>
</tr>
<tr>
<td>415</td>
<td>Unsupported media type</td>
<td>The endpoint does not support the format of the request body.</td>
</tr>
</tbody>
</table>

Dot-walking in REST API requests

You can use dot-walking when specifying the `sysparm_query` or `sysparm_fields` parameters in requests to REST APIs that support those parameters.

Note: The Import Set API does not support dot-walking.
Dot-walking in sysparm_query

You can filter queries using related record values by dot-walking in the `sysparm_query` parameter. For example, you can retrieve all incident records where the incident Company has a specific Stock symbol value.

https://<instance>.service-now.com/api/now/table/incident?
sysparm_query=company.stock_symbol=NYX

Dot-walking in sysparm_fields

You can view field values from multiple tables by dot-walking in the `sysparm_fields` parameter. For example, you can retrieve the Name, Sys_id, and Department of each user that has certain roles, as well as the role Name.

The request runs on the User Roles (sys_user_has_role) table which defines a many-to-many relationship between users and roles. The response includes field values from the User (sys_user) and Roles (sys_user_role) tables.

```
https://<instance>.service-now.com/api/now/table/sys_user_has_role?
sysparm_fields=role%2Crole.name%2Cuser%2Cuser.name%2Cuser.sys_id
%2Cuser.department&sysparm_query=role%3D3d43716d0f6002003a2d47bce1050e0d
%5EOrole%3Dac73b52d0f6002003a2d47bce1050e0e&sysparm_display_value=true
```

```json
{
   "result": [
      {
         "user.name": "Fred Johnson",
         "user.sys_id": "f5a3716d0f6002003a2d47bce1050ed4",
         "role.name": "support",
         "user.department": {
            "display_value": "Accounting",
            "link": "https://<instance>.service-now.com/api/now/table/cmn_department/5b3b13530f58c2003a2d47bce1050e96"
         },
         "role": {
            "display_value": "support",
            "link": "https://<instance>.service-now.com/api/now/table/sys_user_role/3d43716d0f6002003a2d47bce1050e0d"
         },
         "user": {
            "display_value": "Fred Johnson",
            "link": "https://<instance>.service-now.com/api/now/table/sys_user/f5a3716d0f6002003a2d47bce1050ed4"
         }
      },
      {
         "user.name": "Fred Johnson",
         "user.sys_id": "f5a3716d0f6002003a2d47bce1050ed4",
         "role.name": "asset_mgmt",
         "user.department": {
            "display_value": "Accounting",
            "link": "https://<instance>.service-now.com/api/now/table/cmn_department/5b3b13530f58c2003a2d47bce1050e96"
         },
         "role": {
            "display_value": "asset_mgmt",
            "link": "https://<instance>.service-now.com/api/now/table/sys_user_role/ac73b52d0f6002003a2d47bce1050e0e"
         }
      }
   ]
}
```
Debugging REST queries

You can debug REST queries by reviewing the session debug log.

When the `glide.rest.debug` property is `true`, all REST processing is logged in the session debug log.

REST logging includes processing durations, headers, and the request body. Prolonged use of this property can affect performance, so it is best to use it while debugging REST processing, and then set the property back to `false`.

Sample log output

```
2014-03-19 11:10:37 (633) http-12 New transaction 083A6031D7231100261253B28252035C #28 /api/now/table/incident
2014-03-19 11:10:37 (633) REST API-thread-1 SYSTEM DEBUG: [REST API] RESTAPIProcessor : Request Header: user-agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:12.0) Gecko/20100101 Firefox/12.0
2014-03-19 11:10:37 (633) REST API-thread-1 SYSTEM DEBUG: [REST API] RESTAPIProcessor : Request Header: cookie:glide_user_route=glide.20e7f4d6dbc0d44810117aacc0eeae; JSESSIONID=F07CE6ACF8AF2C3AFC237CB239AF43BF360BF; glide_user="U0N2Mjo0MDNhNjAzMWQ3MjMxMTAwMj0yMDMzMTOto2ODE2Zjc5Y2MwYTgwMTY0MDMxMzYyMj0yMj0yMj0yMDMzMTOto2ODE2Zjc5Y2MwYTgwMTY0MDMxMzYyMj0yMj0yMj0yMDMzMTOto2ODE2Zjc5Y2MwYTgwMTY0MDMxMzYyMj0yMj0yMj0yMDMzMTOto2ODE2Zjc5Y2MwYTgwMTY0MDMxMzYyMj0yMj0yMj0yMDMzMTOto2ODE2Zjc5Y2MwYTgwMTY0MDMxMzYyMj0yMj0yMj0yMDMz
2014-03-19 11:10:37 (633) REST API-thread-1 SYSTEM DEBUG: [REST API] RESTAPIProcessor : Route Registry : Loaded Routes to Cache
2014-03-19 11:10:37 (633) REST API-thread-1 SYSTEM DEBUG: [REST API] RouteRegistry : Route loading time 0:00:00.105
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Returning session debug logs in a REST response

You can include session debug logs in a REST response body by passing the X-WantSessionDebugMessages header in the request.

To return session debug messages when session debugging is enabled for the current session, set the header X-WantSessionDebugMessages to true in the REST request.

Note: You must enable session debugging before sending this header.

Request

This example demonstrates a Table API request made using REST API Explorer with Session Debug SQL enabled.

```
GET api/now/table/incident/9c573169c611228700193229fff72400
X-WantSessionDebugMessages:true
Content-Type: application/json;charset=UTF-8
Accept: application/json, text/plain,*/*
X-UserToken: <user token>
```

Response body

```
{
  "result":{
    "description":"User can't access email on mail.company.com.",
    "number":"INC0000001"
  },
  "session":{
    "debug_logs":[
      {
        "type":"sql",
        "customerUpdate":false,
        "line":"17:17:27.777: Time: 0:00:00.000 for: glide_master_db[glide.5] ..... sys_user_session0.‘id’ = '3BEA7001EB32020C46AC2EFEF106FE2A’</span>",
        "debugClassSet":"
      },
      {
        "type":"sql",
        "customerUpdate":false,
        "line":"17:17:27.779: Time: 0:00:00.002 for: glide_master_db[glide.6] ... ‘sys_id’ = '7fead001eb32020c46ac2eef106fe2a’</span>",
        "debugClassSet":"
      }
    ]
  }
}
```
Scripted REST APIs

The Scripted REST APIs feature allows application developers to build web service APIs. You can define service endpoints, query parameters, and headers for a scripted web service API, as well as scripts to manage the request and response.

Scripted REST APIs generally follow the REST architecture, but may be customized to use different conventions.

The following podcast offers additional information on the use of Scripted REST APIs.

The Automated Test Framework supports Inbound REST test steps. You can create automated tests for custom Inbound REST APIs that you create. Creating tests for your custom REST APIs simplifies upgrade testing, and makes it possible to verify modifications to a REST API are backward compatible. See Administering REST test step configurations and ATF REST test step configurations.

Developer training

In the ServiceNow® Developer Site, you can find training for Scripted REST APIs.

Scripted REST core concepts

Review the Scripted REST APIs core concepts before creating your own APIs.

Scripted REST API URIs

Scripted REST API URIs depend on the namespace and API ID, and optionally the version, of the web service.

The API URI, excluding the instance URL, appears in the Resource path and Default resource path fields for scripted REST resource records. The Default resource path field appears only for resources with the default API version.

URIs used by Scripted REST APIs follow these formats:

- https://instance.service-now.com/api/(name_space)/(api_id)/(resource_path)
- https://instance.service-now.com/api/(name_space)/(version)/(api_id)/(resource_path)

In these URIs:

- (name_space) is a read-only value. For web services in the global scope, the name space is the value of the property glide.appcreator.company.code. For web services in a scoped application, the name space is the scope name, such as x_company_apppname.
- (api_id) is the value of the API ID field on the scripted REST API record. By default this value is based on the service name.
- (resource_path) is the relative path defined for the resource. Specifying a relative resource path allows you to have multiple resources using the same HTTP method, such as GET, in one web service. For example, an resource may specify the path /{id} when the web service has only one GET resource, or /user/{id} and /message/{id} when the web service has different resources for requesting user and message records.
- (version) is the version of the API to access if the API uses versioning, such as v1. You can access the default version of a versioned API by using the URI without a version number.
Scripted REST API roles

Certain roles are required to work with scripted REST API records. Roles required to access each API depend on the API implementation.

The following roles are required to work with scripted REST API records.

These roles are not required to access a scripted REST API endpoint. The REST API author can define security requirements for each API and API resource.

<table>
<thead>
<tr>
<th>Role title (name)</th>
<th>Description</th>
<th>Contains Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web service administrator (web_service_admin)</td>
<td>Users with this role can read, create, modify, and delete Scripted REST APIs and web service resources.</td>
<td>none</td>
</tr>
</tbody>
</table>

Scripted REST API query parameters

Query parameters define values that requesting users can pass with a request.

When creating a scripted REST API you can specify which parameters are available and which are mandatory for each request. You can access request parameters in scripted REST API scripts using the request object `queryParams` field.

Scripted REST API versioning

Scripted REST APIs may be versioned, allowing you to test and deploy changes without impacting existing integrations.

Enable versioning

By default, new scripted REST APIs are not versioned. To enable versioning, click Enable versioning on the Scripted REST API form.

Note: To continue supporting non-versioned URLs after enabling versioning, select a version as the default version.

Default version

A version may be marked as default. Specifying a default version allows users to access that version using a web service URL without a version number.

If more than one active version is marked as default, the latest default version is used as the default.

Add a version

To add a new version to a scripted REST service, click Add new version on the Scripted REST API form.

When you add a new version, you can copy resources from an existing version.
Scripted REST API access controls

Scripted REST APIs use REST_Endpoint type ACLs to define access controls.

As a user with the web_service_admin role, you can define access controls for scripted REST APIs and scripted REST API resources.

A scripted REST API or scripted REST resource may require users to submit basic authentication credentials and pass an ACL check. A requesting user must satisfy at least one of the ACLs. It is not necessary to satisfy all selected ACLs.

Scripted REST service ACLs must have the **Type** value **REST_Endpoint**.

Scripted REST API security matrix

There are multiple possible security configurations for scripted REST APIs. Use this table to identify the scripted REST API security configuration that best suits your needs, and the field values to implement that configuration.

**Scripted REST API security**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Scripted REST API</th>
<th>Scripted REST Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default ACLs</td>
<td>Requires authentication</td>
</tr>
<tr>
<td>The resource is public. No authentication or ACL is required.</td>
<td>Any value</td>
<td>False</td>
</tr>
<tr>
<td>The resource requires basic authentication only. No ACL is required.</td>
<td>Any value</td>
<td>True</td>
</tr>
<tr>
<td>The resource requires basic authentication only. No ACL is required.</td>
<td>No ACL selected</td>
<td>True</td>
</tr>
<tr>
<td>An ACL selected in the resource record is required.</td>
<td>Any value</td>
<td>True</td>
</tr>
<tr>
<td>An ACL selected in the scripted REST API record is required.</td>
<td>One or more ACLs selected</td>
<td>True</td>
</tr>
</tbody>
</table>

Scripted REST API error objects

Scripted REST APIs include error objects that allow you to respond to a request with a standard HTTP error message when an error occurs during request processing.

You can use error objects in scripted REST API resources to alert requesting clients of errors. Use error objects to respond to incoming requests, not to catch errors within your server-side code.
**Error response format**

The content type of the response depends on the request Accept header. If the Accept header specifies an unsupported format, such as image/jpeg, the error response uses JSON.

Error responses follow this format:

```json
{
  "error": {
    "message": "My error message",
    "detail": "My details"
  },
  "status": "failure"
}
```

The numeric status code, such as 404, is included in the response Status code header, not in the response body.

**Sending scripted REST API errors**

Scripted REST APIs provide multiple ways to send an error in a response to a requesting client.

The `Script` field on the Scripted REST Resource form provides a `process` function template.

You can send an error response by returning an error object, or using the `process` function response parameter. Either option returns the same error to the requesting client.

**Returning an error**

You can send an error by returning an error object.

```javascript
(function process(request, response) {
  // request is bad
  return new sn_ws_err.BadRequestError('My error message');
})(request,response);
```

**Setting a response error**

You can send an error by calling the `setError` function on the response parameter.

```javascript
(function run(request, response) {
  // request is bad
  responseBuilder.setError(new sn_ws_err.BadRequestError()); // Constructor parameter is optional
})(request,response);
```

**Available error objects**

Multiple error objects are available in scripted REST API scripts to report error information to requesting clients.
Error namespace

All scripted REST API error objects use the sn_ws_err namespace. When instantiating a new error object, use this namespace, such as in `var error = new sn_ws_err.BadRequestError();`.

Predefined errors

Predefined error objects allow you to easily communicate an error based on standard HTTP error codes. You can set an optional error message in the object constructor.

<table>
<thead>
<tr>
<th>Error Object</th>
<th>Description</th>
<th>Status Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadRequestError</td>
<td>Indicates an error in the request, such as incorrect syntax.</td>
<td>400</td>
</tr>
<tr>
<td>NotFoundError</td>
<td>Indicates a requested resource is not available.</td>
<td>404</td>
</tr>
<tr>
<td>NotAcceptableError</td>
<td>Indicates the Accept header value passed in the request is incompatible with the web service.</td>
<td>406</td>
</tr>
<tr>
<td>ConflictError</td>
<td>Indicates that there is a conflict in the request, such as multiple conflicting updates.</td>
<td>409</td>
</tr>
<tr>
<td>UnsupportedMediaTypeError</td>
<td>Indicates the request media type is not supported by the web service.</td>
<td>415</td>
</tr>
</tbody>
</table>

```javascript
(function run(request, response) {
    // request is bad
    return new sn_ws_err.BadRequestError('My error message');
})(request, response);
```

Customizable ServiceError

The ServiceError object allows you to define your own error response. You can set the status code, error message, and error details using a ServiceError object. Use the ServiceError object when none of the predefined errors meet your needs.

The ServiceError object exposes these functions for setting values:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>setStatus(Number status)</td>
<td>Use this function to set the numeric status code for the error.</td>
<td>500</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Default value</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>setMessage(String message)</td>
<td>Use this function to set a short error message.</td>
<td>An empty string</td>
</tr>
<tr>
<td>setDetail(String detail)</td>
<td>Use this function to set the detailed error description.</td>
<td>An empty string</td>
</tr>
</tbody>
</table>

```javascript
(function run(request, response) {
    var myError = new sn_ws_err.ServiceError();
    myError.setStatus(410);
    myError.setMessage('My error Message');
    myError.setDetail('Here are the details about this error');
    return myError;
})(request,response);
```

**Create a Scripted REST API**

Create a Scripted REST API to define web service endpoints.

Role required: web_service_admin

By default, scripted REST APIs contain an ACL that prohibits users with the snc_external role from making requests to the API.

1. Navigate to **System Web Services > Scripted REST APIs**.
2. Click **New**.
3. Enter a **Name** for the service.
   - The **API ID** is set automatically based on the **Name**. You can modify the **API ID** if needed.
4. Click **Submit**.

After you create the API, configure the service as needed such as by creating resources, assigning ACLs, or specifying supported request and response formats.

**Create a scripted REST API resource**

Create a scripted REST API resource to define the HTTP method, the processing script, and to override settings from the parent service.

There must be a Scripted REST API defined before you can create resources.

Role required: web_service_admin

By default, any new Scripted REST API resource you create contains an ACL that prohibits users with the snc_external role from making requests to the API.

1. Navigate to **System Web Services > Scripted REST APIs**.
2. Select a scripted REST API record.
3. In the **Resources** related list, click **New**.
4. Enter a **Name**.
   - The resource name affects the URI for sending requests to the API.
5. Select the **HTTP method** this resource implements, such as **GET**.
6. In the **Script** field, define how the operation parses and responds to requests.
7. Optional: Override settings from the parent REST API as needed, such as by specifying different security settings or valid content types.
8. Optional: On the Documentation tab, provide a Short description explaining how to access the resource. This information appears when exploring this resource using the REST API Explorer.

9. Click Submit.

After creating the resource, you can associate headers and query parameters.

**Define scripted REST API headers**

Define scripted REST API headers to control which headers the API accepts and can respond with. There must be a Scripted REST API defined before you can create headers.

Role required: web_service_admin

1. Navigate to System Web Services > Scripted REST APIs.
2. Select a scripted REST API record.
3. In the Request Headers related list, click New.
4. Enter a Header name.
5. Optional: Select Is required to make this header mandatory for all requests to associated scripted REST resources.
6. Enter a Short description and Example value to explain how to use the header.
7. Click Submit.

After defining available headers, associate the headers with a scripted REST resource.

**Define available query parameters**

Define available query parameters to control what values a requesting user can pass in the request URI.

Role required: web_service_admin

1. Navigate to System Web Services > Scripted REST APIs.
2. Select a scripted REST API record.
3. In the Query Parameters related list, click New.
4. Specify the Query parameter name.
5. Optional: Select Is required to make this parameter mandatory for all requests to associated scripted REST resources.
6. Enter a Short description and Example value to explain how to use the parameter.
7. Click Submit.

After defining available query parameters, associate the parameters with a scripted REST resource.

**Configure a scripted REST API to require an ACL**

Requests to scripted REST APIs respect platform ACLs, and the requesting user must meet any table ACL requirements to access instance data. Additionally, you can configure the scripted REST API to require a specific ACL.

Role required: web_service_admin

The ACLs selected in this task apply to all API endpoints.

1. Navigate to System Web Services > Scripted REST APIs.
2. Select a scripted REST API.
3. In the Default ACLs field, select one or more ACLs that meet the security needs for the API. Select only those ACLs that have a Type of REST_Endpoint. A requesting user must satisfy at least one of the selected ACLs. It is not necessary to satisfy all selected ACLs.
4. Click Update.

You can override the API security settings for each individual resource.

Configure a scripted REST API resource to require an ACL
By default, API resources/endpoints inherit security settings from the parent API. Define custom ACLs for a specific resource/endpoint to override the inherited settings.

Role required: web_service_admin or admin

ACLs are checked for an authenticated user only.

1. Navigate to System Web Services > Scripted REST APIs.
2. Select a scripted REST API.
3. In the Resources related list, select a resource.
4. In the Security tab, select the Requires authentication check box.
   You must select this check box to require an ACL for the resource. If you clear this check box, the resource becomes public and requires no credentials. Clear this check box only if you want to allow unauthenticated requests to access the resource, even if the parent REST service requires an ACL.
5. Select the Requires ACL authorization check box.
6. In the ACL field, select one or more ACLs that meet the security needs for the endpoint. Select only those ACLs that have a Type of REST_Endpoint. Only users who have roles defined in the selected REST_Endpoint type ACL are granted access to this resource.
   Selecting an ACL for a resource overrides any ACLs selected for the parent web service. Leave this field blank to use the ACLs selected for the parent web service.

Enable versioning for a scripted REST API

Enable versioning for a scripted REST API to provide multiple versions of the API while maintaining compatibility with existing integrations.

There must be a Scripted REST API defined before you can enable versioning.

Role required: web_service_admin

1. Navigate to System Web Services > Scripted REST APIs.
2. Select a scripted REST API record.
3. Click Enable versioning.
   The Enable versioning pop up appears. The Make version v1 default check box is selected by default.
4. Optional: Clear the Make version v1 default check box to enable versioning without a default version.
   Versioned APIs without a default version are accessible only by using the version-specific URI. Make version v1 default, or select a different version as default after you enable versioning.
5. Click OK.
   The Versioning embedded list is added to the Scripted REST API form. You can add new versions or control which version is default from this list.

Add a version to a scripted REST API
Add a new version to a versioned scripted REST API to define new API behavior without impacting older versions.
There must be a Scripted REST Service that has versioning enabled before you can add a new version.

Role required: web_service_admin

1. Navigate to System Web Services > Scripted REST APIs.
2. Select a scripted REST service record.
3. Click Add new version.
   The Add new version pop up appears.
4. Optional: Select Make this version the default to configure the REST service to use the new version as the default version.
5. Optional: In the Copy existing resources from version choice list, select an existing version to copy all resources from that version to the new version.
6. Click OK.

Controlling request and response content type

You can control which content types are allowed in scripted REST API requests and responses.

You can set default values for the service using the Default supported request formats and Default supported response formats fields. These fields define acceptable values users can pass in the Content-Type and Accept request headers respectively. If a requesting user specifies an Accept or Content-Type header not supported by the service or resource, the instance responds with an HTTP error code 406 or 415.

You can override these values for each resource using the Supported request formats and Supported response formats on the Scripted REST Resource form.

Note: The Supported request formats field only appears for PUT, POST, and PATCH resources.

You can use wildcard values when specifying valid content types.

If no accepted content types are specified, scripted REST web services support Application/json, Application/xml, and Text/xml by default.

Important: If the request body format is not Application/json, Application/xml, or Text/xml, use only the request body dataStream field to access the request body. Using request body data, dataString, nextEntry(), or hasNext() with a non-default format will result in a 500 error response.

When sending a binary type in a response, you must set the response content type and write the binary stream directly using a RESTAPIResponseStream object. You can access this object by calling getStreamWriter() on the response object.

Controlling maximum request size

You can specify the maximum file size allowed in a scripted REST API request payload.

The file size limit applies when accessing any of the following variables or functions from a RESTAPIRequestBody object.

- data
- dataString
- nextEntry()
- hasNext()

Accessing these variables or functions with a request payload that exceeds the maximum size causes the service to respond with error code 400.
The file size limit does not apply when accessing the `dataStream` variable.

**Maximum request size properties**

Several properties control the maximum allowed request size. Add these properties to the System Properties (`sys_properties`) table to specify a maximum request size.

**Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>glide.rest.scripted.max_inbound_content_length_mb</code></td>
<td>10</td>
<td>The maximum size, in megabytes, for a scripted REST request body that is not gzipped.</td>
</tr>
<tr>
<td><code>glide.rest.scripted.max_inbound_gzip_content_length_mb</code></td>
<td>1</td>
<td>The maximum size, in megabytes, for a scripted REST request body that is gzipped.</td>
</tr>
</tbody>
</table>
| `glide.rest.max_content_length`              | 10            | The maximum size, in megabytes, for a scripted REST request body, whether or not it is gzipped.  
  - Maximum: 25 |

As a result, even if `glide.rest.scripted.max_inbound_content_length_mb` or `glide.rest.scripted.max_inbound_gzip_content_length_mb` are set, the request body is limited to the value of `glide.rest.max_content_length`.

---

**Explore and document Scripted REST APIs**

You can browse scripted REST APIs using the REST API Explorer and view documentation provided by scripted REST API creators.

The REST API must be active and must specify at least one resource to appear in the REST API Explorer.

You can provide additional information about scripted REST APIs, API resources, request headers, and query parameters. The **Short description** field from these records appears on the REST API Explorer when viewing the appropriate API and resource. You can also provide a **Documentation link** URL in the scripted REST API record to direct users to more detailed documentation. Users can access this URL by clicking **API documentation** in the REST API Explorer.

**Scripted REST APIs good practices**

Follow these guidelines when designing and implementing Scripted REST APIs.

**Follow REST API conventions**

Use REST API standards to provide a consistent and easy to use interface for clients.
REST API conventions define specific behavior for each type of method. Use the following guidelines as a starting point for designing your API.

- **GET** operations only query data. A GET request should never modify data.
- **POST** operations create new records but do not modify existing records.
- **PUT** and **PATCH** operations modify existing records.
- **DELETE** operations destroy records.

**Use versioning to control changes to your API**

Use versioning to implement new functionality and avoid breaking existing integrations.

When you introduce significant functionality changes to your API, create a new version of the API first. Do not introduce behavior that will break existing integrations in a published version.

Using versioning allows you to implement significant changes to your API without breaking existing clients. You can then release the new version of the API for new clients while allowing existing clients to upgrade at their own pace.

Encourage clients to use a version-specific API, or configure the API without a default version to force clients to specify a version. You can also make new, optional behavior available by adding an optional parameter to an existing version.

**Return an informative HTTP status code**

Return a status code that informs the requestor of the success or failure of the request.

Return an HTTP status code that helps the client understand the result of the request. Use the following guidelines for common status codes.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Indicates that the request was completed successfully.</td>
</tr>
<tr>
<td>201</td>
<td>Indicates that a record was created successfully.</td>
</tr>
<tr>
<td>204</td>
<td>Indicates that a record was deleted successfully.</td>
</tr>
<tr>
<td>40X (401, 404)</td>
<td>Status codes in the 400 range indicate a client error, such as 400 for invalid request syntax.</td>
</tr>
<tr>
<td>50X (500, 503)</td>
<td>Status codes in the 500 range indicate that a server error occurred. The client request may have been valid or invalid, but a problem occurred on the server that prevented it from processing the request.</td>
</tr>
</tbody>
</table>

**Return useful error information**

Provide the client with enough information in error messages to allow them to understand the problem without having to refer to your API documentation.

An error response should include a helpful error message, as well as an error status code.

For example, when a client queries a record that does not exist, you can return the error message "The specified record does not exist. Ensure that a record with the ID of <id value> exists in the application." along with a 404 status code.
The Scripted REST APIs feature includes several preconfigured error objects you can use for commonly-encountered errors, and a customizable ServiceRequest error object you can use when the preconfigured error objects do not meet your needs.

Enforce and test access controls

Enforce existing access controls and require additional access to modify data.

In addition to requiring authentication to access the API, require authorization to access data. Use the GlideRecordSecure API in your Scripted REST API scripts. This API ensures that access controls defined on the underlying data are applied for the requesting user.

Require additional access controls for operations that modify data. Requests such as PUT, POST, and DELETE should require a higher level of access than GET. Configure these API resources to require a more strict ACL.

Test your access controls, both authentication and authorization, before releasing the API.

Build tests to verify functionality

Build tests that verify your Scripted REST web services functionality as part of your development process.

Use repeatable tests to ensure that your API functions the way you expect it to. Testing also helps ensure that changes you make do not affect the expected API behavior after you release a version. You can use a REST client application that supports automated testing, such as Postman, to facilitate testing.

Tests should validate the response code, headers, and body content as appropriate for each resource you implement. You can also use tests to validate authentication requirements, and to confirm that errors return useful responses.

RESTAPIRequest

A RESTAPIRequest object allows you to access scripted REST API request details in scripts.

Note: You cannot instantiate objects of this type. Objects of this type are created automatically and are accessible only in scripted REST API resource scripts.

RESTAPIRequest - body

The body of the request.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>body</td>
<td>RESTAPIRequestBody</td>
<td>The body of the request. You can access data from the body object using the RESTAPIRequestBody API.</td>
</tr>
</tbody>
</table>

var requestBody = request.body // Returns instance of RESTAPIRequestBody
**RESTAPIRequest - pathParams**

The path parameters passed in the request URI.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pathParams</td>
<td>Object</td>
<td>The path parameters as a script object. Available path parameters depend on the web service configuration.</td>
</tr>
</tbody>
</table>

In this example, the scripted REST API endpoint follows this format: https://instance.service-now.com/api/now/myservice/{tableName}/{id}. The request being processed uses this URL: https://instance.service-now.com/api/now/myservice/myApp_table/1234.

```javascript
var pathParams = request.pathParams;
var tableName = pathParams.tableName; // ‘myApp_table’
var id = pathParams.id; // ‘1234’
```

**RESTAPIRequest - queryParams**

The query parameters from the web service request.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryParams</td>
<td>Object</td>
<td>The query parameters from the web service request.</td>
</tr>
</tbody>
</table>

In this example, the request being processed uses this URL: https://<instance_rest_endpoint>?active=false&name=now. Note the active and name parameters.

```javascript
var queryParams = request.queryParams;
var isActiveQuery = queryParams.active; // false
var nameQueryVal = queryParams.name; // ‘now’
```

**RESTAPIRequest - queryString**

The entire query added to the endpoint URI.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryString</td>
<td>String</td>
<td>The entire query for the request.</td>
</tr>
</tbody>
</table>
In this example, the request being processed uses this URL: https://<instance_rest_endpoint>?active=false&name=now. Note the query active=false&name=now.

```javascript
var query = request.queryString; //"active=false&name=now"
```

**RESTAPIRequest - uri**

The request URI, excluding domain information.

<table>
<thead>
<tr>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>url</td>
</tr>
</tbody>
</table>

In this example, the request being processed uses this URL: https://instance.service-now.com/api/now/table/myTable?active=false&name=now.

```javascript
var query = request.uri; //"api/now/table/myTable"
```

**RESTAPIRequest - url**

The entire request URL.

<table>
<thead>
<tr>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>url</td>
</tr>
</tbody>
</table>

In this example, the request being processed uses this URL: https://instance.service-now.com/api/now/table/myTable?active=false&name=now.

```javascript
var query = request.url; //"https://instance.service-now.com/api/now/table/myTable?active=false&name=now"
```

**RESTAPIRequest - headers**

All headers from the request.

<table>
<thead>
<tr>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>headers</td>
</tr>
</tbody>
</table>
var headers = request.headers;
var acceptHeader = headers.Accept;
var myCustomHeader = headers.myCustom;
var specialHeader = headers['special - header'];

**RESTAPIRequest - getHeader(String header)**

Returns the value of a specific header from the web service request.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>String</td>
<td>The name of the header, such as accept or content-type.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>The value of the specified header.</td>
</tr>
</tbody>
</table>

var acceptHeader = request.getHeader('accept');

**RESTAPIRequest - getSupportedResponseContentTypes()**

Get the content types specified in the request Accept header.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array of string values where each string is a content type, such as application/json.</td>
</tr>
</tbody>
</table>

**RESTAPIRequestBody**

A RESTAPIRequestBody object allows you to access the body content of a scripted REST API request in scripts.

The format of a RESTAPIRequestBody object may be JSON or XML, depending on the content-type header value from the request.
**Note:** You cannot instantiate objects of this type. Objects of this type are created automatically and are accessible only in scripted REST API resource scripts.

Single entry example-request-body in JSON format.

```json
{
    "name": "user1",
    "id": 1234,
    "roles": [
        {
            "name": "admin"
        },
        {
            "name": "itil"
        }
    ]
}
```

Multiple entry example-request-body in JSON format.

```json
[
    {
        "name": "user1",
        "id": 1234,
        "roles": [
            {
                "name": "admin"
            },
            {
                "name": "itil"
            }
        ]
    },
    {
        "name": "user2",
        "id": 9876,
        "roles": [
            {
                "name": "admin"
            }
        ]
    }
]
```

**Important:** If the request body format is not Application/json, Application/xml, or Text/xml, use only the request body `dataStream` field to access the request body. Using request body `data`, `dataString`, `nextEntry()`, or `hasNext()` with a non-default format will result in a 500 error response.

**RESTAPIRequestBody - data**

The content of the request body.

**Note:** REST web services consume `data`, `datastream`, and `dataString` via stream. Because streams are only consumed once, only one can be called. For example, if calling both `data` and `dataString`, the second call will return empty.
### RESTAPIRequestBody - data

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Object or Array</td>
<td>The request content. This can be a single object or an array of objects depending on the request.</td>
</tr>
</tbody>
</table>

```javascript
var entry;
var id;
var requestBody = request.body;
var requestData = requestBody.data; // May be an array or a single object
if (requestData instanceof Array) {
    entry = requestData[0].name; // 'user1'
    id = requestData[0].id; // '1234'
} else {
    entry = requestData.name; // 'user1'
    id = requestData.id; // '1234'
}
```

### RESTAPIRequestBody - dataStream

The content of the request body, as a stream.

**Note:** REST web services consume data, datastream, and dataString via stream. Because streams are only consumed once, only one can be called. For example, if calling both data and dataString, the second call will return empty.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataStream</td>
<td>Object</td>
<td>The content of the request body. You can pass the stream to a separate API, such as to create an attachment from the request or forward the request to a different endpoint.</td>
</tr>
</tbody>
</table>

```javascript
var requestBody = request.body;
var requestStream = requestBody.dataStream;
```

### RESTAPIRequestBody - dataString

The content of the request body, as a String.

**Note:** REST web services consume data, datastream, and dataString via stream. Because streams are only consumed once, only one can be called. For example, if calling both data and dataString, the second call will return empty.
### Field

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataString</td>
<td>String</td>
<td>The content of the request body.</td>
</tr>
</tbody>
</table>

```javascript
var requestBody = request.body;
var requestString = requestBody.dataString;
```

### RESTAPIRequestBody - hasNext()

Determine if there are additional entries in the request body.

Use this method with the nextEntry() method to iterate over multiple request body entries.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>None</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>True if there are additional entries available. This method returns true only once if the request contains a single entry.</td>
</tr>
</tbody>
</table>

```javascript
var requestBody = request.body;
requestBody.hasNext(); // returns true if the request contains a single entry or multiple entries

// calling second time
requestBody.hasNext(); // returns false if the request contains a single entry, or true if the request contains multiple entries
```

### RESTAPIRequestBody - nextEntry()

Retrieve one entry from the request body as a script object.

Use this method with the hasNext() method to iterate over multiple request body entries.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>None</td>
</tr>
</tbody>
</table>
### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>A single entry from the request body.</td>
</tr>
</tbody>
</table>

```javascript
var requestBody = request.body;
var requestEntry = requestBody.nextEntry(); // returns available entry if there is only one entry, or the first entry if there are multiple.
var name = requestEntry.name; // ‘user1’

// Calling second time
requestEntry = requestBody.nextEntry(); // returns undefined if there is only one entry, or the second entry if there are multiple.
```

This example demonstrates using hasNext() with nextEntry().

```javascript
var requestBody = request.body;
while(requestBody.hasNext()){
  var entry = requestBody.nextEntry();
}
```

### RESTAPIResponse

A RESTAPIResponse object allows you to build a RESTful response to a scripted REST API request.

**Note:** You cannot instantiate objects of this type. Objects of this type are created automatically and are accessible only in scripted REST API resource scripts.

### RESTAPIResponse - getStreamWriter()

Get the ResponseStreamWriter for this response, allowing you to write directly to the response stream.

Set the content type and status code using the setHeaders and setStatus functions prior to calling the getStreamWriter function.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RESTAPIResponseStream</code></td>
<td>The ResponseStreamWriter for this response. You can use this object to write directly to the response stream.</td>
</tr>
</tbody>
</table>

```javascript
response.setContentType('application/json');
response.setStatus(200);
var writer = response.getOutputStream();
```

### RESTAPIResponse - `setBody(Object body)`

Sets the body content to send in the web service response.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>body</td>
<td>Object</td>
<td>The response body, as a JavaScript object. The body content is automatically serialized to JSON or XML depending on the value of the <code>Accept</code> header passed in the request.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var body = {};
body.name = "incident";
body.number = "1234";
body.caller = {"id": "user1"};
response.setBody(body);
```

```javascript
var bodyArray = [];
var body = {};
body.name = "incident";
body.number = "1234";
body.caller = {"id": "user1"};
bodyArray.push(body);
response.setBody(bodyArray);
```
RESTAPIResponse - setHeaders(Object headers)
Sets the headers for the web service response.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headers</td>
<td>Object</td>
<td>A JavaScript object listing each header and the value to assign that header.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
var headers = {};
headers.X-Total-Count=100;
headers.Location='https://instance.service-now.com/<endpoint_to_resource>,'
response.setHeaders(headers);
```

RESTAPIResponse - setLocation(String location)
Assigns a value to the Location header in the web service response.
See the **W3 Location header documentation** for more information about this header.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>String</td>
<td>An absolute URI to redirect the response recipient to.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

RESTAPIResponse - setStatus(Number status)
Sets the status code number for the web service response.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Number</td>
<td>The status code to send in the response, such as 200 to indicate success. Passing a non-numerical value, such as a string, causes the status code to default to 0.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
response.setStatus(200);
```

### RESTAPIResponse - setHeader(String header, String value)

Assign a value to a REST service response header.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>String</td>
<td>The header you want to set.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value to assign the specified header.</td>
</tr>
</tbody>
</table>

#### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```java
responseBuilder.setHeader("Location","<URI>);
```

### RESTAPIResponse - setContentType(String contentType)

Assigns a value to the Content-Type header in the web service response.

You must set a response content type before writing the response. The content type is set automatically for string responses, based on the request Accept header value.

Setting an invalid content type causes the response to default to JSON. Failing to set a content type results in a status code 500 error when sending a binary response.

See the W3 Content-Type header documentation for more information about this header.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contentType</td>
<td>String</td>
<td>The content type of the response body, such as application/json.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
responseBuilder.setContentType('application/json');
```

`RESTAPIResponse - setError(Object error)`

Configure the response to return an error.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>Object</td>
<td>An error object.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

`RESTAPIResponseStream`

A `RESTAPIResponseStream` object allows you to write directly to the scripted REST API response stream.

Use `RESTAPIResponseStream` methods to build web service APIs in the `Scripted REST APIs` feature.

**Note:** You cannot instantiate objects of this type. Objects of this type are created automatically and are accessible only in scripted REST API resource scripts.

`RESTAPIResponseStream - writeStream(Object stream)`

Write an input stream to the response stream.

You must set the content type and status code before calling the `writeStream()` method or the response will fail. You cannot modify these values after calling the `writeStream()` method.

**Note:**
It is the responsibility of the script author to obtain the stream from a third-party service.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream</td>
<td>Object</td>
<td>An attachment or a response stream from a third-party service.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

The following example is for scoped applications:

```javascript
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    response.setContentType('application/json');
    response.setStatus(200);

    var gsa = new GlideSysAttachment();
    var attachmentStream = new gsa.getContentStream(<sys_id of attachment>);
    var writer = response.getOutputStreamWriter();
    writer.writeStream(attachmentStream);

})(request, response);
```

The following example is for global applications:

```javascript
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    response.setContentType('application/json');
    response.setStatus(200);

    var attachmentStream = new GlideSysAttachmentInputStream(<sys_id of attachment>);
    var writer = response.getOutputStreamWriter();
    writer.writeStream(attachmentStream);

})(request, response);
```

**RESTAPIResponseStream - writeString(String data)**

Write string data to the response stream.

You must set the content type and status code before calling the writeString() method or the response will fail. You cannot modify these values after calling the writeString() method.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>String</td>
<td>The string to add to the response data.</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
response.setContentType('application/json');
response.setStatus(200);
var writer = response.getStreamWriter();
var body ={  
    name:user1,
    id: 1234,
    roles: [
        {  
            name: admin
        },
        {  
            name: itil
        }
    ]
}
writer.writeString("{'name':'user','id':'1234'}");
writer.writeString(JSON.stringify(body));
```

### Scripted REST API examples

Multiple examples are available demonstrating how to create and use scripted REST APIs.

**Scripted REST API example - script samples**

These examples demonstrate how to create various resource scripts for a scripted REST API.

**Query parameters GET example**

This example demonstrates how to get query parameter values from a request.

```javascript
/**
 * GET - Sample Request API - Query Params
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    var uri = request.uri;
    var url = request.url;
    var queryParams = request.queryParams;
    var customHeader = request.getHeader('X-Custom');

    return {
        "uri": uri,
```
Path parameters GET example

This example demonstrates how to get path parameter values from a request.

```javascript
/**
 * GET - Sample Request API - Path Params
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
  var uri = request.uri;
  var url = request.url;
  var path = request.pathParams;

  return {
    "uri": uri,
    "url": url,
    "path_params": path,
    "path.id": path.id
  }
})(request, response);
```

Script include GET example

This example demonstrates how to use a script include to provide a response. By using a script include you can reuse common code and maintain readability in the REST service scripts.

```javascript
/**
 * GET - Sample Request API - Script Include
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
  var responseObj = global.SampleDataUtil.getSampleJSON();
  return responseObj;
})(request, response);
```

String POST example

This example demonstrates how to parse a POST message with a string body and send a response based on the request.

```javascript
/**
 * POST - Sample Request API - dataString
 * sample usage:
 * var requestBody = request.body;
 * var requestString = requestBody.dataString;
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
  var requestBody = request.body;
```
var requestString = requestBody.dataString;
                return {"requestString": requestString};
            })(request, response);

**Binary POST example**

This example demonstrates how to parse a POST message with a binary body and send a response based on the request.

```javascript
/**
 * POST - Sample Request API - Body
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    var body = request.body.data;
    // do any additional processing on the request body, such as inserting a new record.
    return {
        "body.id": body.id
    };
})(request, response);
```

**Not acceptable error example**

This example demonstrates how to respond with a not acceptable error. Use this error type when the request Accept header value is not supported by the web service.

```javascript
/**
 * Sample Not Acceptable Error Sample
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    response.setError(new sn_ws_err.NotAcceptableError('sample error message'));
})(request, response);
```

**Bad request error example**

This example demonstrates how to respond with a bad request error. Use this error type to indicate a mistake in the request syntax.

```javascript
/**
 * Bad Request Error Sample
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    response.setError(new sn_ws_err.BadRequestError('sample error message'));
})(request, response);
```
Conflict error example

This example demonstrates how to respond with a conflict error. Use this error type in the event of multiple conflicting requests, such as multiple updates to the same record.

```javascript
/**
 * Error Response: Conflict Error Sample
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    response.setError(new sn_ws_err.ConflictError('sample error message'));
})(request, response);
```

Not found error example

This example demonstrates how to respond with a not found error. Use this error type if the requested resource does not exist or is unavailable.

```javascript
/**
 * Error Response: Not Found Error Sample
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    response.setError(new sn_ws_err.NotFoundError('sample error message'));
})(request, response);
```

Unsupported media type error example

This example demonstrates how to respond with an unsupported media type error. Use this error type to indicate that the Content-Type of the request is unsupported.

```javascript
/**
 * Error Response: Unsupported Media Type Error Sample
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    response.setError(new sn_ws_err.UnsupportedMediaTypeError('sample error message'));
})(request, response);
```

Service error example

This example demonstrates how to respond with a generic service error. The ServiceError object allows you to define the status code, message, and error detail. Use a ServiceError if the predefined error types do not meet your needs.

```javascript
/**
 * Error Response: Custom Error Sample
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    var myError = new sn_ws_err.ServiceError();
    myError.setStatus(418);
    myError.setMessage("I am a Teapot");
});
```
myError.setDetail("Here are the details about this error");
response.setError(myError);
}(request, response);

Scripted REST resource script example

This sample REST API resource script parses the name and id values from the request body and returns those values in the response.

```javascript
/**
 * POST - Sample Request API - Body
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
  var body = request.body.data,
      id0,name0,id1,name1;
  name0 = body[0].name; // 'user0'
  id0 = body[0].id; // '1234'
  name1 = body[1].name; // 'user1'
  id1 = body[1].id; // '5678'

  return {
    "id": id0,
    "name": name0,
    "id": id1,
    "name": name1
  };
})(request, response);

Requests

The API can accept both XML and JSON requests.

<table>
<thead>
<tr>
<th>JSON Request</th>
<th>XML Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST /api/sn_demo_api/v1/example/body HTTP/1.1</td>
<td>POST /api/sn_demo_api/v1/example/body HTTP/1.1</td>
</tr>
<tr>
<td>Content-Type: application/json</td>
<td>Content-Type: application/json</td>
</tr>
<tr>
<td>Accept: application/json</td>
<td>Accept: application/json</td>
</tr>
<tr>
<td>Host: &lt;instance&gt;.service-now.com</td>
<td>Host: &lt;instance&gt;.service-now.com</td>
</tr>
<tr>
<td>Connection: close</td>
<td>Connection: close</td>
</tr>
<tr>
<td>Content-Length: 91</td>
<td>Content-Length: 152</td>
</tr>
</tbody>
</table>
| [                                                  | <request><entry>
|   {                                                 |   <name>user0</name>
|     "name": "user0",                               |     <id>1234</id>
|     "id": 1234                                    |   </entry>
|   },                                                | <entry>
|   {                                                 |   <name>user1</name>
|     "name": "user1",                               |     <id>5678</id>
|     "id": 5678                                    |   </entry>
| }                                                  | </request>                                       |
Responses

Both requests specify application/json as the Accept header value. This causes either response to use JSON formatting, even if the request content type is XML.

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 04 Aug 2015 15:20:44 GMT
Server: ServiceNow
Connection: close
Set-Cookie: BIGIpServerpool_<Instance>=880838154.47166.0000; path=/

{"result":{"id":1234,"id1":5678,"name":"user0","name1":"user1"}}

Scripted REST API example - streaming vs object serialization

These examples demonstrate how to send a JSON response using streaming and using default object serialization.

Streaming vs object serialization

When sending a response, you can send a response as a stream or serialize an object. There are advantages and disadvantages to either approach. Pick a technique based on the needs of your integration.

Generally, if the response object is simple, can be represented as XML or JSON, and is a consistent size, use object serialization. If using a format other than XML or JSON, or if the size of the response varies, use streaming.

Streaming the response

Using a streaming responses provides advantages in response time, instance performance, and content flexibility, but adds additional complexity to the script. When using streaming, you are responsible for formatting the response, setting the response status, and setting the Content-Type header. When streaming a response, the requesting user receives a response quickly because the entire response does not need to be created before starting streaming.

This example demonstrates a Scripted REST Resource script that returns an array of incident records using streaming.

```javascript
/**
 * Sample Scripted REST Resource that returns custom JSON objects with properties from Incident GlideRecords
 * This sample uses ServiceNow JavaScript API to query incident records
 * and then iterates over those records to build and stream a custom JSON object that
 * includes some values from the incidents
 */
(function runOperation(/*RESTServiceRequest*/ request, /*RESTServiceResult*/ response) {
  var writer = response.getOutputStream(),
  hdrs = {},
  table = 'incident',
  record_limit = 100,
  ...
gr = new GlideRecord(table);

hdrs['Content-Type'] = 'application/json';
response.setStatus(200);
response.setHeaders(hdrs);

gr.setLimit(record_limit);
gr.query();

// start building response object
writer.writeString("{"results":[]");

// iterate over incident records and build JSON representations to be streamed out.
while (gr.next()) {
    var incidentObj = {};
    incidentObj.number = gr.number + '';
    incidentObj.short_description = gr.short_description + '';
    writer.writeString(global.JSON.stringify(incidentObj));
    if (gr.hasNext()) {
        writer.writeString(",");
    }
}

// close the response object
writer.writeString("}");
}(request, response);

A request to this resource returns the following response.

// sample response
/*
HTTP/1.1 200 OK
Content-Type: application/json
Server: ServiceNow

// sample response number of records returned has been truncated for this example
[
"results": [
{"number": "INC0011301",
"short_description": "lorem ipsum short description 0 my new incident"},
{"number": "INC0011302",
"short_description": "lorem ipsum short description 1 my new incident"},
{"number": "INC0011303",
"short_description": "lorem ipsum short description 2 my new incident"},
{"number": "INC0011304",
"short_description": "lorem ipsum short description 3 my new incident"}
],
*/
Building an object

Using object serialization allows you to take advantage of ServiceNow provided serialization and content negotiation. When serializing an object instead of streaming, the entire object must be created and serialized before the client receives a response. This may delay the response, or require a large amount of system resources if the response object is very large. Object serialization is available only for XML or JSON responses. Responses using a different format must use streaming.
This example returns the same Incident data as the streaming example, but collects all of the response data in an array before sending the response.

```javascript
/**
 * Sample Scripted REST Resource returns an array of custom JSON objects that include 2 incident properties.
 * This sample uses ServiceNow JavaScript API to query incident records and then iterates
 * over those records building a custom JSON object that includes 2 values from the incident records.
 * note that because we are returning a simple JSON object we can rely on built in serialization
 * to set the content-type header as well as response status.
 * The 'result_arr' object will not be returned
 * until it has been completely built and stored
 */
(function runOperation(/*RESTServiceRequest*/ request, /*RESTServiceResult*/ response) {
  var table = 'incident',
      record_limit = 100,
      result_arr = [],
      gr = new GlideRecord(table);

  gr.setLimit(record_limit);
  gr.query();

  // iterate over incident records and build JSON representations to be streamed out.
  while (gr.next()) {
    var incidentObj = {};
    incidentObj.number = gr.number + '';
    incidentObj.short_description = gr.short_description + '';
    result_arr.push(incidentObj);
  }

  return result_arr;
})(request, response);
```

A request to this resource returns the following response.

```plaintext
/*
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Server: ServiceNow

// sample response number of records returned has been truncated for this example

{
  "result": [{
    "short_description": "lorem ipsum short description 0 my new incident",
    "number": "INC0011301"
  }, {
    "short_description": "lorem ipsum short description 1 my new incident"
  }]
*/
```
Scripted REST API example - streaming file attachments

This example demonstrates how to send an image attachment to a requesting user as a binary stream.

```javascript
/**
 * Sample Scripted REST Resource that returns a stream of binary representing an attachment
 * This sample uses ServiceNow JavaScript API GlideSysAttachmentInputStream to get an attachment as a stream then
 * users WriteStream to stream the response.
 */
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
  var hdrs = {},
      attachment_sys_id = '1852fd52471321009db4b5b08b9a71a9';

  hdrs['Content-Type'] = 'image/jpeg';
  response.setStatus(200);
  response.setHeaders(hdrs);

  var writer = response.getWriter();
})
```
var attachmentStream = new GlideSysAttachmentInputStream(attachment_sys_id);
writer.writeStream(attachmentStream);
})(request, response);

A request to this resource returns the following response.

```html
/*
   // sample response
   HTTP/1.1 200 OK
   Set-Cookie: glide_session_store=SYSTEM; Expires=Fri, 30-Oct-2015 21:57:00 GMT; Path=/; HttpOnly
   Content-Type: image/jpeg
   Transfer-Encoding: chunked
   Date: Fri, 30 Oct 2015 21:26:59 GMT
   Connection: close
   Server: ServiceNow

   <binary response body excluded from this sample>
*/
```

**SOAP web service**

The SOAP Web Services provided by ServiceNow are WS-I compliant, as outlined in the WS-I Basic Profile 1.0.

**Concepts and Terminology**

- **Provider** - publishes web service for clients to invoke (consume)
- **Consumer** - invokes / consumes published web service
- **Standards**
  - **WSDL**
    - Web Service Description Language
    - XML document describing functions, arguments, data schema, and endpoint (where / how to invoke the service, URL)
    - WSDL only necessary when generating SOAP envelope programmatically
  - **SOAP**
    - Simple Object Access Protocol
    - XML document usually HTTP posted to web service endpoint described in WSDL
    - SOAP:Envelope / SOAP:Header / SOAP:Body
  - **HTTP**
    - Hyper-Text Transfer Protocol
    - POST vs GET - Web Service is POSTed
    - XML vs. Form POST - Web Service is XML
Web Service Provider

ServiceNow publishes its underlying table structures and associated data with the following Web Service methods:

- Direct Web Services: Use a URL query to request a ServiceNow table's WSDL.
- Web Service Import Sets: Use import tables and transform maps to automate Web Service requests to ServiceNow tables.
- Scripted Web Services: Use custom JavaScript to execute Web Services requests.

Note: SOAP messages are sent with the assumption that the recipient is XML compliant. No encoding is applied to the SOAP message. SOAP always decodes responses as UTF-8, the XML encoding header is not used.

SOAP direct web service API functions

The standard SOAP API is a set of small, globally defined functions that can be performed on a targeted resource.

The targeted resource (or table) is defined in the URL by the format `https://<instance name>.service-now.com/<table name>.do`.

Data Retrieval API

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getKeys</td>
<td>Query the targeted table by example values and return a comma delimited sys_id list.</td>
</tr>
<tr>
<td>getRecords</td>
<td>Query the targeted table by example values and return all matching records and their fields.</td>
</tr>
<tr>
<td>get</td>
<td>Query a single record from the targeted table by sys_id and return the record and its fields.</td>
</tr>
<tr>
<td>aggregate</td>
<td>Query using and aggregate functions SUM, COUNT MIN, MAX and AVG. To enable the aggregate functions, activate the Aggregate Web Service Plugin.</td>
</tr>
</tbody>
</table>

Data Modification API

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert</td>
<td>Creates a new record for the table targeted in the URL.</td>
</tr>
<tr>
<td>insertMultiple</td>
<td>Creates multiple new records for the table targeted in the URL. To enable multiple inserts, activate the Web Service Insert Multiple Plugin.</td>
</tr>
<tr>
<td>update</td>
<td>Updates a existing record in the targeted table in the URL, identified by the mandatory sys_id field.</td>
</tr>
<tr>
<td>deleteRecord</td>
<td>Deletes a record from the targeted table by supplying its sys_id.</td>
</tr>
<tr>
<td>Method Summary</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>deleteMultiple</td>
<td>Delete multiple records from the targeted table by example values.</td>
</tr>
</tbody>
</table>

**Data Retrieval API**

Data Retrieval API method summaries and descriptions.

### Data Retrieval API

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getKeys</td>
<td>Query the targeted table by example values and return a comma delimited sys_id list.</td>
</tr>
<tr>
<td>getRecords</td>
<td>Query the targeted table by example values and return all matching records and their fields.</td>
</tr>
<tr>
<td>get</td>
<td>Query a single record from the targeted table by sys_id and return the record and its fields.</td>
</tr>
<tr>
<td>aggregate</td>
<td>Query using and aggregate functions SUM, COUNT MIN, MAX and AVG. To enable the aggregate functions, activate the Aggregate Web Service Plugin.</td>
</tr>
</tbody>
</table>

**getKeys**

Query the targeted table by example values and return a comma delimited sys_id list.

**Input fields**

Any field value in the targeted table.

**Output fields**

A SOAP response element sys_id that contains a comma delimited list of sys_id values.

**Sample SOAP messages**

Sample SOAP request

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <inc:getKeys>
      <category>hardware</category>
    </inc:getKeys>
  </soapenv:Body>
</soapenv:Envelope>
```
Sample SOAP response

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <getKeysResponse>
      <sys_id>46e18c0fa9fe19810066a0083f76bd56,46e57642a9fe1981000b96a5dca501ff,46f1784ba9fe19810018aa27fbb23482</sys_id>
      <count>7</count>
    </getKeysResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Language-specific sample messages

For language-specific `getKeys` samples, refer to the following topics:

- **Perl SOAP::Lite**
- **Java Apache Axis2**
- **Python**

`getRecords`  
Query the targeted table by example values and return all matching records and their fields.

**Input fields**

Any field value in the targeted table.

**Output fields**

The `getRecordResponse` element may contain one or more `getRecordsResult` elements that encapsulate elements representing the field values of records matching the query.

Sample SOAP messages

Sample SOAP request

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <inc:getRecords>
      <number>INC00000002</number>
    </inc:getRecords>
  </soapenv:Body>
</soapenv:Envelope>
```
Sample SOAP request using an encoded query to filter where incident number is INC0000001 or INC0000002

```xml
    <soapenv:Header/>
    <soapenv:Body>
        <inc:getRecords>
            <!--encoded_query-->number=INC0000001^ORnumber=INC0000002</encoded_query>
        </inc:getRecords>
    </soapenv:Body>
</soapenv:Envelope>
```

Sample SOAP response that contains 1 record

```xml
    <soapenv:Header/>
    <soapenv:Body>
        <getRecordsResponse>
            <getRecordsResult>
                <caller_id>5137153cc611227c000bbd1bd8cd2007</caller_id>
                <caller_id.email>david.loo@service-now.com</caller_id.email>
                <closed_at/>
                <number>INC0000002</number>
                <opened_at>2009-12-14 23:07:12</opened_at>
                <short_description>Can't get to network file shares</short_description>
            </getRecordsResult>
        </getRecordsResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Sample SOAP response that contains more than 1 record

```xml
    <soapenv:Header/>
    <soapenv:Body>
        <getRecordsResponse>
            <getRecordsResult>
                <caller_id>5137153cc611227c000bbd1bd8cd2006</caller_id>
                <caller_id.email>rick.berzle@yourcompany.com</caller_id.email>
                <closed_at>2009-12-17 22:55:16</closed_at>
                <number>INC0000009</number>
                <opened_at>2009-12-16 22:50:23</opened_at>
                <short_description>Reset my password</short_description>
            </getRecordsResult>
        </getRecordsResponse>
    </soapenv:Body>
</soapenv:Envelope>
```
Language-specific sample messages

For language-specific `getRecords` samples, refer to the following topics:

- **Perl SOAP::Lite**
- **Java Apache Axis2**
- **Python**

**get**

Query a single record from the targeted table by `sys_id` and return the record and its fields.

**Input fields**

An element `<sys_id>` identifying the `sys_id` of the record to be retrieved.

**Output fields**

A `getResponse` element encapsulating all field values for the record retrieved.

**Sample SOAP messages**

**Sample SOAP request**

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <inc:get>
      <sys_id>46e18c0fa9fe19810066a0083f76bd56</sys_id>
    </inc:get>
  </soapenv:Body>
</soapenv:Envelope>
```
The resulting response of a `get` function call looks like this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <getResponse xmlns="http://www.service-now.com/incident">
      <active>1</active>
      <approval>not requested</approval>
      <assigned_to>46c6f9efa9fe198101ddf5eed9adf6e7</assigned_to>
      <caller_id>46b673a6a9fe19810007ab03cbd5849d</caller_id>
      <category>network</category>
      <cmdb_ci>0c43f35dc61122750182c132a29e3243</cmdb_ci>
      <comments>Testing</comments>
      <contact_type>phone</contact_type>
      <due_date>2007-10-28 13:29:45</due_date>
      <escalation>0</escalation>
      <impact>3</impact>
      <incident_state>1</incident_state>
      <knowledge>0</knowledge>
      <location>1081761cc611227501d063fd3475a2de</location>
      <made_sla>1</made_sla>
      <notify>1</notify>
      <number>INC10055</number>
      <opened_at>2007-09-18 00:32:09</opened_at>
      <opened_by>46bac3d6a9fe19810005f299d979b8869</opened_by>
      <priority>0</priority>
      <reassignment_count>0</reassignment_count>
      <severity>0</severity>
    </getResponse>
  </soap:Body>
</soap:Envelope>
```

**aggregate**

Query a table using an `aggregate` function including SUM, COUNT, MIN, MAX, and AVG.

**Note:** Functionality described here requires the Aggregate Web Service plugin.

**Input fields**

Any element of the target table. In addition, one or more of the `aggregate` functions (SUM, COUNT, MIN, MAX, and AVG).

A `GROUP BY` and a `HAVING` clause may also be added.

**Output fields**

An `aggregateResponse` element encapsulating all field values for the record retrieved.
Sample SOAP messages

Sample SOAP request using COUNT aggregate function.

```xml
<?xml version="1.0" encoding="UTF-8"?>
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:m="http://www.service-now.com"
  xmlns:tns="http://www.service-now.com/map"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <aggregate>
      <COUNT>number</COUNT>
      <active>true</active>
    </aggregate>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The resulting response of a COUNT aggregate function call looks like this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:m="http://www.service-now.com"
  xmlns:tns="http://www.service-now.com/map"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <aggregateResponse>
      <aggregateResult>
        <avg>2.7200</avg>
      </aggregateResult>
    </aggregateResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample SOAP request using AVG aggregate function with a GROUP BY clause.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:m="http://www.service-now.com"
  xmlns:tns="http://www.service-now.com/map"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <aggregateResponse>
      <aggregateResult>
        <avg>2.7200</avg>
      </aggregateResult>
    </aggregateResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
The resulting response of a AVG aggregate function call with a GROUP BY clause looks like this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Body>
    <aggregateResponse>
      <aggregateResult>
        <avg>1.0000</avg>
        <category>database</category>
      </aggregateResult>
      <aggregateResult>
        <avg>3.0000</avg>
        <category>hardware</category>
      </aggregateResult>
      <aggregateResult>
        <avg>3.0000</avg>
        <category>inquiry</category>
      </aggregateResult>
      <aggregateResult>
        <avg>2.0000</avg>
        <category>network</category>
      </aggregateResult>
      <aggregateResult>
        <avg>2.6923</avg>
        <category>software</category>
      </aggregateResult>
    </aggregateResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Sample SOAP request using an encoded query to filter the aggregate:

```xml
<?xml version="1.0" encoding="UTF-8"?>
xmlns:m="http://www.service-now.com"
xmlns:tns="http://www.service-now.com/map"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <aggregate>
      <COUNT>number</COUNT>
      <active>true</active>
      <__encoded_query>number=INC0000001^ORnumber=INC0000002</__encoded_query>
    </aggregate>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample aggregate request using HAVING to narrow the results.

HAVING takes four fields. Each field is delimited by "^": the aggregate type, the field of the aggregate, the operation type, and the value to compare.

More than one HAVING can be added to the request, so you can use HAVING expressions, but there is no support for OR.

```xml
<?xml version="1.0" encoding="UTF-8"?>
xmlns:m="http://www.service-now.com"
xmlns:tns="http://www.service-now.com/map"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <aggregate>
      <COUNT>sys_id</COUNT>
      <GROUP_BY>internal_type</GROUP_BY>
      <HAVING>COUNT^*^>^10</HAVING>
      <HAVING>COUNT^*^<^20</HAVING>
      <COUNT>sys_id</COUNT>
      <active>true</active>
    </aggregate>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Data Modification API

Data Modification API method summaries and descriptions.

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert</td>
<td>Creates a new record for the table targeted in the URL.</td>
</tr>
<tr>
<td>insertMultiple</td>
<td>Creates multiple new records for the table targeted in the URL. To enable multiple inserts, activate the Web Service Insert Multiple Plugin.</td>
</tr>
<tr>
<td>update</td>
<td>Updates a existing record in the targeted table in the URL, identified by the mandatory sys_id field.</td>
</tr>
<tr>
<td>deleteRecord</td>
<td>Deletes a record from the targeted table by supplying its sys_id.</td>
</tr>
<tr>
<td>deleteMultiple</td>
<td>Delete multiple records from the targeted table by example values.</td>
</tr>
</tbody>
</table>

**insert**

Creates a new record for the table targeted in the URL.

**Input fields**

All fields from the targeted table, excluding system fields. Fields configured as mandatory in the System Dictionary are reflected in the WSDL with the attribute minOccurs=1.

**Output fields**

**Insert method output fields**

<table>
<thead>
<tr>
<th>Table type</th>
<th>Output fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular</td>
<td>The sys_id field and the display value of the target table (table) are returned.</td>
</tr>
</tbody>
</table>
**Table type** | **Output fields**
---|---
Import set | The **sys_id** of the import set row, the name of the transformed target table (**table**), the **display_name** for the transformed target table, the **display_value** of the transformed target row, and a **status** field, which can contain **inserted**, **updated**, or **error**.

There can be an optional **status_message** field or an **error_message** field value when **status=error**.

When an insert did not cause a target row to be transformed (skipped because a key value is not specified), the **sys_id** field will contain the **sys_id** of the import set row, rather than the targeted transform table.

Import set with multiple transforms | The response from this type of insert will contain multiple sets of fields from the regular import set table insert wrapped in a **multiInsertResponse** parent element. Each set will contain a **map** field, showing which transform map created the response.

---

**Sample SOAP messages for a regular table**

The following example shows an insert that specifies the short description only:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
                   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
                   xmlns:m="http://www.service-now.com"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <insert xmlns="http://www.service-now.com">
      <short_description xsi:type="xsd:string">This is a test</short_description>
    </insert>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The resulting response looks like this:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
                   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
                   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <!-- Response content -->
</SOAP-ENV:Envelope>
```
Language-specific sample messages

For language-specific insert samples, refer to the following topics:

Perl SOAP::Lite
Java Apache Axis2
Microsoft .NET
Python

insertMultiple
Creates multiple new records for the table targeted in the URL.

Input fields

The insertMultiple element may contain 1 or more record tags that contains all fields from the targeted table, excluding system fields. Limit the number of records inserted in a single operation to no more than 200. You can gradually increase this number with subsequent exports if the increase does not negatively impact instance performance.

Output fields

The insertMultipleResponse tag is followed by 1 or more record tags that contains:

<table>
<thead>
<tr>
<th>Table type</th>
<th>Output fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular</td>
<td>The sys_id field and the display value of the target table (table) are returned.</td>
</tr>
</tbody>
</table>
### Table type

<table>
<thead>
<tr>
<th>Output fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import set</td>
</tr>
<tr>
<td>Import set with multiple transforms</td>
</tr>
</tbody>
</table>

### Sample SOAP messages for a regular table

The following example shows an insert that specifies the short description only:

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <inc:insertMultiple>
      <record>
        <short_description>this is test 1</short_description>
      </record>
      <record>
        <short_description>this is test 2</short_description>
      </record>
      <record>
        <short_description>this is test 3</short_description>
      </record>
    </inc:insertMultiple>
  </soapenv:Body>
</soapenv:Envelope>
```

The resulting response looks like this:

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <inc:insertMultipleResponse>
      <recordResponse>
        <record><short_description>this is test 1</short_description></record>
        <record><short_description>this is test 2</short_description></record>
        <record><short_description>this is test 3</short_description></record>
      </recordResponse>
    </inc:insertMultipleResponse>
  </soapenv:Body>
</soapenv:Envelope>
```
Sample SOAP messages for an import set table

The following example shows an insert that specifies the short description only:

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <imp:insertMultiple>
      <imp:record>
        <imp:message>one</imp:message>
        <imp:uuid>a</imp:uuid>
      </imp:record>
      <imp:record>
        <imp:message>two</imp:message>
        <imp:uuid>b</imp:uuid>
      </imp:record>
      <imp:record>
        <imp:message>three</imp:message>
        <imp:uuid>c</imp:uuid>
      </imp:record>
    </imp:insertMultiple>
  </soapenv:Body>
</soapenv:Envelope>
```

The resulting response looks like this:

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <insertMultipleResponse>
      <insertResponse>
        <sys_id>1296b3ab0a0a0b5b73e966f6fab7acde</sys_id>
      </insertResponse>
    </insertMultipleResponse>
  </soapenv:Body>
</soapenv:Envelope>
```
**update**

Updates an existing record in the targeted table in the URL, identified by the mandatory **sys_id** field.

**Input fields**

All fields from the targeted table, excluding system fields, which will be used for updating the existing record. The **sys_id** field is used to locate the existing record.

**Output fields**

Returns the **sys_id** of the record that was updated.

**Sample SOAP messages**

**Sample SOAP request**

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <inc:update>
      <sys_id>46e18c0fa9fe19810066a0083f76bd56</sys_id>
      <short_description>this is updated</short_description>
    </inc:update>
  </soapenv:Body>
</soapenv:Envelope>
```
Sample SOAP response

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <updateResponse>
      <sys_id>46e18c0fa9fe19810066a0083f76bd56</sys_id>
    </updateResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Language-specific sample messages

For language-specific update samples, refer to the following topics:

- **Perl SOAP::Lite**
- **Java Apache Axis2**
- **Microsoft .NET**
- **Python**

**deleteRecord**
Delete a record from the targeted table by supplying its `sys_id`.

**Input fields**

An element `<sys_id>` identifying the `sys_id` of the record to be retrieved.

**Output fields**

A `<count>` element within the `deleteRecordResponse` parent element indicating the number of records deleted, this will always equal to '1' for `deleteRecord`.

**Sample SOAP messages**

Sample SOAP request

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <inc:deleteRecord>
      <sys_id>46e18c0fa9fe19810066a0083f76bd56</sys_id>
    </inc:deleteRecord>
  </soapenv:Body>
</soapenv:Envelope>
```
Sample SOAP response

```xml
   >
   <soapenv:Header/>
   <soapenv:Body>
      <deleteRecordResponse>
         <count>1</count>
      </deleteRecordResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Language-specific sample messages

For language-specific `deleteRecord` samples, refer to the following topics:

- **Perl SOAP::Lite**
- **Java Apache Axis2**
- **Microsoft .NET**
- **Python**

`deleteMultiple`

Delete multiple records from the targeted table by example values.

**Input fields**

All fields from the targeted table, including system fields, are used in query-by-example (QBE) fashion to locate records to be deleted. Query example fields can have special prefixes to constrain the search function.

**Output fields**

A `<count>` element within the `deleteRecordResponse` parent element indicating the number of records deleted.

Sample SOAP messages

Sample SOAP request

```xml
   <soapenv:Header/>
   <soapenv:Body>
      <inc:deleteMultiple>
         <category>hardware</category>
      </inc:deleteMultiple>
   </soapenv:Body>
</soapenv:Envelope>
```
Sample SOAP response

```xml
  <soapenv:Header/>
  <soapenv:Body>
    <deleteMultipleResponse>
      <count>6</count>
    </deleteMultipleResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Language-specific sample messages

For language-specific deleteRecord samples, refer to the following topics:

- Perl SOAP::Lite
- Java Apache Axis2
- Microsoft .NET
- Python

WSDL

All ServiceNow tables and import sets dynamically generate WSDL XML documents that describe its table schema and available operations.

You can get a WSDL format by issuing a URL targeting a ServiceNow table with the WSDL parameter, for example:

```
https://myinstance.service-now.com/incident.do?WSDL
```

All dynamically generated and served ServiceNow WSDL accessible via HTTP is available for use under the terms defined in the Open Source Initiative OSI - Apache License, Version 2.0 license agreement.

Apache License, Version 2.0

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.
"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.


Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make,
have made, use, offer to sell, sell, import, and otherwise transfer the
Work, where such license applies only to those patent claims licensable by
such Contributor that are necessarily infringed by their Contribution(s)
alone or by combination of their Contribution(s) with the Work to which
such Contribution(s) was submitted. If You institute patent litigation
against any entity (including a cross-claim or counterclaim in a lawsuit)
alleging that the Work or a Contribution incorporated within the Work
constitutes direct or contributory patent infringement, then any patent
licenses granted to You under this License for that Work shall terminate as
of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works
thereof in any medium, with or without modifications, and in Source or
Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a
copy of this License; and
2. You must cause any modified files to carry prominent notices stating
that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You
distribute, all copyright, patent, trademark, and attribution notices from
the Source form of the Work, excluding those notices that do not pertain to
any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution,
then any Derivative Works that You distribute must include a readable copy
of the attribution notices contained within such NOTICE file, excluding
those notices that do not pertain to any part of the Derivative Works, in
at least one of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or documentation,
if provided along with the Derivative Works; or, within a display generated
by the Derivative Works, if and wherever such third-party notices normally
appear. The contents of the NOTICE file are for informational purposes only
and do not modify the License. You may add Your own attribution notices
within Derivative Works that You distribute, alongside or as an addendum
to the NOTICE text from the Work, provided that such additional attribution
notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may
provide additional or different license terms and conditions for use,
reproduction, or distribution of Your modifications, or for any such
Derivative Works as a whole, provided Your use, reproduction, and
distribution of the Work otherwise complies with the conditions stated in
this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally
submitted for inclusion in the Work by You to the Licensor shall be under
the terms and conditions of this License, without any additional terms or
conditions. Notwithstanding the above, nothing herein shall supersede or
modify the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks,
service marks, or product names of the Licensor, except as required for
reasonable and customary use in describing the origin of the Work and
reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.
Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

Direct web services

A direct web service is available for any table in the system provided the correct access control is setup.

The supported format of the incoming message is document style literal XML SOAP documents (Document/Literal). To retrieve the direct web service WSDL description and XML schema, point to the relative URL of <tablename>.do?WSDL. For example, to retrieve the WSDL for the Incident table on the online demo system, use the following URL:

https://<instance name>.service-now.com/incident.do?WSDL

Using forms to limit or extend the query response

On occasion, there is a need to limit the number of field values being returned from a SOAP query (get or getRecords) invocation.

Specifying a form view has the effects of:

1. limiting the response elements to contain only the fields on the view
2. specifying reference record field values from referenced fields eg. caller_id.email, this will cause the value of the caller's email to be returned in the SOAP response

To enable form views for SOAP queries, you may configure the property com.glide.soap.view to be the name of the view you wish to use for all SOAP query response (the default issoap_response). You may also specify the view name as a URL parameter sysparm_view=<view name> when making the SOAP call, for example:
If a specified view name does not exist, the default behavior is to respond with all fields.

### Direct web services extended query parameters

The following parameters, when specified as elements of input parameters to SOAP query functions such as get, getKeys, and getRecords, has the additional behavior of filtering and modifying the results that are returned.

**Note:** Extended query element names are preceded by two underscore characters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>__encoded_query</td>
<td>Specify an encoded query string to be used in filtering the returned results. The encoded query string format is similar to the value that may be specified in sysparm_query URL parameter. You may refer to the encoded query building example in the RSS feed generator examples.</td>
<td><code>&lt;__encoded_query&gt;active=true^category='hardware'&lt;/__encoded_query&gt;</code></td>
</tr>
<tr>
<td>__order_by</td>
<td>Instruct the returned results to be ordered by the specified field</td>
<td><code>&lt;__order_by&gt;priority&lt;/__order_by&gt;</code></td>
</tr>
<tr>
<td>__order_by_desc</td>
<td>Instruct the returned results to be ordered by the specified field, in descending order</td>
<td><code>&lt;__order_by_desc&gt;opened_date&lt;/__order_by_desc&gt;</code></td>
</tr>
<tr>
<td>__exclude_columns</td>
<td>Specify a list of comma delimited field names to exclude from the result set</td>
<td><code>&lt;__exclude_columns&gt;sys_created_on,sys_created_by,caller_id,priority&lt;/__exclude_columns&gt;</code></td>
</tr>
<tr>
<td>__limit</td>
<td>Limit the number of records that are returned</td>
<td><code>&lt;__limit&gt;100&lt;/__limit&gt;</code></td>
</tr>
<tr>
<td>__first_row</td>
<td>Instruct the results to be offset by this number of records from the beginning of the set. When used with __last_row has the effect of querying for a window of results. The results are inclusive of the first row number.</td>
<td><code>&lt;__first_row&gt;250&lt;/__first_row&gt;</code></td>
</tr>
<tr>
<td>__last_row</td>
<td>Instruct the results to be limited by this number of records from the beginning of the set, or the __start_row value when specified. When used with __first_row has the effect of querying for a window of results. The results are less than the last row number, and does not include the last row.</td>
<td><code>&lt;__last_row&gt;500&lt;/__last_row&gt;</code></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>__use_view</td>
<td>Specify a Form view by name, to be used for limiting and expanding the results returned. When the form view contains deep referenced fields eg. caller_id.email, this field will be returned in the result as well</td>
<td><code>&lt;__use_view&gt;soap_view&lt;/__use_view&gt;</code></td>
</tr>
</tbody>
</table>

**Return display value for reference variables**

When you query a record using a get or getRecords function the instance returns all fields associated with that record. The fields are often reference fields that contain a sys_id for a record on another table.

Use one of these options if you want the display value for the field to be returned instead of the sys_id:

1. Add the property glide.soap.return_displayValue to your system properties, and every SOAP request will return a display value for a reference field.

2. Add the parameter `displayvalue=true` to your SOAP request URL, and SOAP requests with that parameter will return a display value for a reference field as a string, instead of the sys_id. The SOAP URL would look as follows: `https://<instance name>.service-now.com/incident.do?displayvalue=true&SOAP`

3. Add the parameter `displayvalue=all` to your SOAP request URL, and SOAP requests with that parameter will return a display value for a reference field, in addition to the sys_id. The response element name for the display value field will be prefixed with `dv_` such as `dv_caller_id`.

**Retrieving journal entries using direct web services**

To get the contents of a journal field, make a second soap request against the sys_journal_field table to pull the appropriate journal records back for the record in question.

The URL for the WSDL would be in the following format.

`https://instance-name.service-now.com/sys_journal_field.do?WSDL`

To retrieve the journal entries, you will first need to query the incident for its `sys_id` value and then supply it as the `element_id` value in a `getRecords` call. To specify records only for the comments field, specify the value comments for the element field. For example, a SOAP request would look like the following.

```xml
 >
 <soapenv:Header />
 <soapenv:Body>
  <sys:getRecords>
   <element>comments</element>
   <element_id>9d385017c611228701d22104cc95c371</element_id>
  </sys:getRecords>
 </soapenv:Body>
</soapenv:Envelope>
```

**Retrieving choice fields using direct web services**

To retrieve or set choice fields, use the choice Value not the Label.

For example, if you want to retrieve a list of all closed incidents use the numerical value for Closed, which is 7 by default.

```xml
<state>7</state>
```

To see a list of choice values:

1. Navigate to the form containing the choice field. For example, navigate to Incident > Open and select an incident.
2. Right-click the choice value field and select **Configure Dictionary** (**Personalize Dictionary** in versions prior to **Fuji**). For example, configure the dictionary for the **State** field.

3. From the **Choices** related list, note the value for the label you want to query. For example, note that the **Closed** choice has a value of 7.

### Choices Table

<table>
<thead>
<tr>
<th>Table</th>
<th>Element</th>
<th>Language</th>
<th>Value</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>incident</td>
<td>state</td>
<td>en</td>
<td>1</td>
<td>New</td>
</tr>
<tr>
<td>incident</td>
<td>state</td>
<td>en</td>
<td>2</td>
<td>Active</td>
</tr>
<tr>
<td>incident</td>
<td>state</td>
<td>en</td>
<td>3</td>
<td>Awaiting Problem</td>
</tr>
<tr>
<td>incident</td>
<td>state</td>
<td>en</td>
<td>4</td>
<td>Awaiting User Info</td>
</tr>
<tr>
<td>incident</td>
<td>state</td>
<td>en</td>
<td>5</td>
<td>Awaiting Evidence</td>
</tr>
<tr>
<td>incident</td>
<td>state</td>
<td>en</td>
<td>6</td>
<td>Resolved</td>
</tr>
<tr>
<td>incident</td>
<td>state</td>
<td>en</td>
<td>7</td>
<td>Closed</td>
</tr>
<tr>
<td>problem</td>
<td>state</td>
<td>en</td>
<td>1</td>
<td>Open</td>
</tr>
<tr>
<td>problem</td>
<td>state</td>
<td>en</td>
<td>3</td>
<td>Pending Change</td>
</tr>
<tr>
<td>problem</td>
<td>state</td>
<td>en</td>
<td>2</td>
<td>Known Error</td>
</tr>
<tr>
<td>problem</td>
<td>state</td>
<td>en</td>
<td>4</td>
<td>Closed/Resolved</td>
</tr>
<tr>
<td>task</td>
<td>state</td>
<td>en</td>
<td>-5</td>
<td>Pending</td>
</tr>
<tr>
<td>task</td>
<td>state</td>
<td>en</td>
<td>1</td>
<td>Open</td>
</tr>
<tr>
<td>task</td>
<td>state</td>
<td>en</td>
<td>2</td>
<td>Work in Progress</td>
</tr>
<tr>
<td>task</td>
<td>state</td>
<td>en</td>
<td>3</td>
<td>Closed Complete</td>
</tr>
<tr>
<td>task</td>
<td>state</td>
<td>en</td>
<td>4</td>
<td>Closed Incomplete</td>
</tr>
<tr>
<td>task</td>
<td>state</td>
<td>en</td>
<td>7</td>
<td>Closed Skipped</td>
</tr>
</tbody>
</table>

**Persisting an HTTP session across all SOAP calls**

In circumstances when a SOAP client makes many calls in a short amount of time, you may want to re-use a single HTTP session for all SOAP calls.

Each SOAP call creates a new user session that persists until it expires. To create a single user session and re-use it for all inbound SOAP calls, develop your SOAP client following these guidelines:

- Use a module like HTTP::Cookies to create a "cookie jar".
- Save the cookies returned by ServiceNow after each request (handled automatically by HTTP::Cookies).
- Re-send the cookies in the cookie jar with each subsequent request.

**Note:** If you have enabled the session rotation high security setting, it will immediately invalidate the JSESSIONID returned from the server with the first response header. The second response includes a new JSESSIONID.

In perl, you can automatically save and send cookies with the following code:

```perl
use HTTP::Cookies;

# we want to store and re-send cookies
my $cookies = HTTP::Cookies->new(ignore_discard => 1);

my $soap = SOAP::Lite

# Set the cookie jar
$soap->transport->cookie_jar($cookies);
```

**Compatibility for clients generated from WSDL**

Review these guidelines for service namespaces.

**Specifying a Unique Namespace for each Table**

The property glide.wsdl.definition.use_unique_namespace ensures that each table's Direct Web Service WSDL has a unique targetNamespace attribute. This property is true by default, which requires a table's Direct Web Service WSDL to use a targetNamespace value of http://www.service-now.com/<table name>. When false (or when the property is not present), all tables use the same targetNamespace value of http://www.service-now.com. Since all tables also share the same operation names, a Web Service client attempting to consume more than one ServiceNow Web Service would be unable to differentiate between requests between multiple tables. Using a unique targetNamespace value allows Web Services clients to distinguish requests between multiple tables.

For example, the Direct Web Service WSDL for the incident table uses this targetNamespace value.

```xml
<wSDL:types><xsd:schema elementFormDefault = "unqualified" targetNamespace = "http://www.service-now.com/incident" />
```

**Setting Namespace Requirements**

ServiceNow’s WSDL schema by default declares an attribute of elementFormDefault=“unqualified”. This attribute indicates whether or not locally declared elements must be qualified by the target namespace in an instance document. If the value of this attribute is 'unqualified', then locally declared elements should not be qualified by the target
namespace. If the value of this attribute is ‘qualified’, then locally declared elements must be qualified by the target namespace.

However, this is incompatible with the way clients generated from WSDL (.NET, Axis2, webMethods, etc.) process the embedded schema, it removes the schema namespace as a result, making the web service response unparseable.

To overcome this compatibility issue, a boolean property called `glide.wsdl.schema.UnqualifiedElementFormDefault` is introduced. This property has the value of `true` by default, setting it to `false` will make clients generated from WSDL able to parse the return value of the web service invocation. You can modify this property using the Web Services properties page at System Properties > Web Services.

### Allowing Duplicate Service Names

By default, service names from dynamically generated WSDL are unique and have the following format:

```
ServiceNow_<table name>
```

To allow duplicate service names, administrators can set the `glide.wsdl.unique_service_name` property to `false`. Create the property if it does not exist.

### Direct web services Perl example - querying all incidents 5 records at a time

This example query queries all incidents, orders by the number field, and retrieves the first 5 records.

You must have the soap_admin role.

You can easily extend this example to retrieve a set of predefined records, `n` number of records each query, simulating a windowed querying client. Using a windowed query mechanisms overcomes the default limitation of only getting a maximum of 250 records per query.

```perl
#!/usr/bin/perl -w

# use SOAP::Lite ( +trace =>; all, maptype => {} );
use SOAP::Lite;

# basic auth using the ITIL user
sub SOAP::Transport::HTTP::Client::get_basic_credentials {
  return 'itil' => 'itil';
}

# specify the endpoint to connect
my $soap = SOAP::Lite
  -> proxy('https://<instance name>.service-now.com/incident.do?SOAP');

my $method = SOAP::Data->name('getRecords')
  ->attr({xmlns => 'http://www.service-now.com/'});

# get all incidents with a window of 5, starting at row 0, and less than row 5 (total of 5 records)
my @params = ( SOAP::Data->name(__first_row => '0') );
push(@params, SOAP::Data->name(__last_row => '5') );
# the last row number can also be taken as the 'limit' offset by the starting first row
push(@params, SOAP::Data->name(__order_by => 'number') );
my $result = $soap->call($method => @params);
```
SOAP web service import sets

Web Service Import Sets complement Direct Web Services and Scripted Web Services to provide a web service interface to Import Sets tables.

This type of web service will transform the incoming data synchronously based on the associated Transform Maps by default.

SOAP web services security

ServiceNow enforces web service security using a combination of basic authentication challenge/response over the HTTPS protocol and system-level access control using contextual security.

Administrators can control what system resources web services users can access by granting them one of the SOAP roles.

SOAP roles

To use SOAP web services, you must have the appropriate role for the operation you want to perform.

Additionally, you must have any other roles required to access the target tables.
SOAP Roles

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>soap</td>
<td>Can perform all SOAP operations.</td>
</tr>
<tr>
<td>soap_create</td>
<td>Can insert new records.</td>
</tr>
<tr>
<td>soap_delete</td>
<td>Can delete existing records.</td>
</tr>
<tr>
<td>soap_ecc</td>
<td>Can query, insert, and delete records on the Queues [ecc_queue] table.</td>
</tr>
<tr>
<td>soap_query</td>
<td>Can query record information.</td>
</tr>
<tr>
<td>soap_query_update</td>
<td>Can query record information and update records.</td>
</tr>
<tr>
<td>soap_script</td>
<td>Can run scripts that specify a .do endpoint. This role is required for running Scripted Web Services.</td>
</tr>
<tr>
<td>soap_update</td>
<td>Can update records.</td>
</tr>
<tr>
<td>import_admin</td>
<td>Can manage all aspects of Import Sets and imports. Required for access to the sys_import_set_row table.</td>
</tr>
<tr>
<td>import_transformer</td>
<td>Can manage Import Set Transform Maps and run transforms. Required for access to the sys_import_set_row table.</td>
</tr>
</tbody>
</table>

Default web services role requirements

By default, a set of processor access control list (ACL) rules require users to have the soap role in order to make WSDL, XSD, and XML Schema requests.

If you want change these role requirements, you can deactivate the ACL rules.

Web service processor ACLs

Configure SOAP security

Administrators can configure web service security for inbound SOAP requests made to the ServiceNow instance.
Role required: admin

You can also set up web service security to use different certificates for different web service clients. By enabling web service security, you can prevent man-in-the-middle attacks.

**Note:** After you configure a WS-security profile or a security policy, validation is performed on all incoming SOAP requests, including from the MID Server or ODBC driver. Disable validation for these types of requests by marking the service accounts as internal integration users.

1. Upload a certificate to the instance.
2. Create a WS-security profile.
3. Create a security policy.

Security policies define which WS-security profiles are used to evaluate a particular web service request. If no policy is defined, all WS-security profiles are used to evaluate all requests.

4. Set the value of the property `glide.soap.default_security_policy` to the name of the new security policy.

**Basic authentication**

Enforce basic authentication for web services to require a username and password with a web service request.

To enforce basic authentication for the user associated with the instance for each WSDL or SOAP message request, administrators can set the property `glide.basicauth.required` to `true`.

When enabled, each WSDL and SOAP request must contain an 'Authorization' header as specified in the Basic Authentication protocol.

Because web services requests are non-interactive, ServiceNow always requires the Authorization header during a request.

**Note:** Basic Authentication refers to local credentials or LDAP authentication, if configured.

Supplying basic authentication information with every request (whether or not it is required) has the added advantage that ServiceNow can associate web service invocations with the user supplied in the basic authentication credentials. For example, when creating an Incident record, the journal fields lists the user ID contained in the basic authentication header instead of the default Guest user.

**Basic authentication code samples**

Samples of basic authentication code for several programming languages and versions.

**Perl and the SOAP::Lite libraries**

To supply basic authentication when using Perl and the SOAP::Lite libraries, you can implement the following function:

```perl
sub SOAP::Transport::HTTP::Client::get_basic_credentials {
    return 'user_name' => 'password';
}
```
C# .NET VS 2005 or older

When using C# .NET VS 2005 or older, you can take advantage of the Credentials object. For example:

```csharp

service.ServiceNow proxy = new service.ServiceNow();
service.get getService = newservice.get();
service.getResponse getServiceResponse = new service.getResponse();

try
{
    proxy.Credentials = cred;
    getService.sys_id = "bf522c350a0a140701972dbf876f1610";
    getServiceResponse = proxy.get(getService);
}

catch (Exception ex) {}  
```

C# .NET VS 2008

When using C# .NET VS 2008, you can take advantage of the ClientCredentials object. For example:

```csharp
Demo_Incident.ServiceNowSoapClient client = new Test08WebService.Demo_Incident.ServiceNowSoapClient();
client.ClientCredentials.UserName.UserName = "admin";
client.ClientCredentials.UserName.Password = "admin";

Then in your app.config file look for the following and change "None" to "Basic":

```xml
<transport clientCredentialType="None" proxyCredentialType="None" realm="" />
```

VB .NET

When using VB .NET taking advantage of the Credentials object would look like the following:

```vbnet
Sub Main()
    Dim cred As New System.Net.NetworkCredential("user_name", "password")
    Dim proxy As New VB_Democm.incident.ServiceNow
    Dim getIncident As New VB_Democm.incident.get
    Dim getResponse As New VB_Democm.incident.getResponse

    proxy.Credentials = cred
    getIncident.sys_id = "[your sysID here]"
    getResponse = proxy.get(getIncident)

End Sub
```
The resulting response when Basic Authentication is turned on and no credentials are supplied looks like this:

```html
<html>
<head>
<title>Apache Tomcat/5.0.28 - Error report</title>
<style>
   H1 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:22px;}
   H2 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:16px;}
   H3 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:14px;}
   BODY {font-family:Tahoma,Arial,sans-serif;color:black;background-color:white;}
   B {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;}
   P {font-family:Tahoma,Arial,sans-serif;background:white;color:black;font-size:12px;}
   A {color: black;}
   A {color: black;}
   HR {color: #525D76;-->
</style>
</head>
<body>
<h1>HTTP Status 401 -
</h1>
<hr size="1" noshade="noshade">
<p><b>type</b> Status report</p>
<p><b>message</b> <u></u></p>
<p><b>description</b>
This request requires HTTP authentication ().</p>
<hr size="1" noshade="noshade">
<h3>Apache Tomcat/5.0.28</h3>
</body>
</html>
```

### WS-Security

Validate signed web services requests with WS-security.

ServiceNow supports **WS-Security 1.1** to validate signed web services requests. Enable WS-Security to:

- Verify SOAP messages originate from a known sender
- Verify SOAP messages have not been altered in transit

**Note:** ServiceNow does not use WS-Security as an encryption mechanism. ServiceNow relies on the HTTPS protocol to encrypt all communications.

WS-Security is intended to work in conjunction with **basic authentication**. When ServiceNow receives a SOAP message, it reviews the basic authentication header to determine if the SOAP user has rights to the instance. It reviews the WS-Security header to determine the validity of the incoming message. Requests affected by attacks such as a man-in-the-middle attack have an invalid WS-Security header and are blocked.

**Enable WS-Security verification**

Administrators can enable Web Services Security (WSS) verification from the Web Services system properties.

Role required: web_service_admin or admin

1. Navigate to **System Web Services > Properties**.
2. For **Require WS-Security header verification for all incoming SOAP requests**, select **Yes**.

   ![Boolean checkbox]

   **Note:** Selecting this option enables WS-Security for all inbound SOAP requests. It is not possible to enable WS-Security for only some requests.

3. Click **Save**.
4. Create a **WS-security profile**.
5. Update the user record for the **MID Server and ODBC driver** to **mark these users as internal integration users**.
6. Download and install the latest **MID Server and ODBC driver**.
7. To validate SOAP request signatures, upload the remote web service's certificate as a JKS and create the web service's WSS Username Token Profile.

   **Note:** Because ServiceNow's WSS implementation does not verify the CA certificate, you do not need to upload the web service's CA certificate.

---

Mark service accounts as internal integration users

Allow internal integration communications to bypass the WSS authentication requirement by marking their user accounts as internal integration users.

**Role required:** admin

When WS-Security is enabled, authentication is required for all SOAP requests including internal integration communications such as the MID Server, ODBC Driver, Remote Update Sets, and high availability cloning. SOAP requests for these internal integration communications cannot implement WS-Security due to technical implications. If your instance uses these SOAP interfaces, you can allow them to bypass the WS-Security authentication requirement by marking their user accounts as internal integration users.

   **Note:** Any web services other than ODBC, MID Server, Remote Update Sets, or high availability cloning must implement WS-Security headers when WS-Security is enabled.

1. Navigate to **User Administration > Users**.
2. Select the user account for the MID Server or ODBC Driver.
3. Configure the form to add the **Internal Integration User** field.
4. Select the **Internal Integration User** check box.
5. Click **Update**.

---

**WS-Security profiles**

A WS-security profile determines how ServiceNow authenticates a web services message when WS-security is enabled.

ServiceNow can authenticate web services requests with the following mechanisms:
### Web service authentication mechanisms

<table>
<thead>
<tr>
<th>Authentication mechanism</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate verification</td>
<td>ServiceNow authenticates the web services request by verifying the certificate associated with the request. Verifying the request's certificate requires uploading the requestor's certificate and certificate authority.</td>
</tr>
<tr>
<td>User credentials</td>
<td>ServiceNow authenticates the Web Services request by verifying the user credentials associated with the request. ServiceNow can either verify that the request's credentials match an existing ServiceNow user's credentials or that the request's credentials match a username and password provided in the profile record.</td>
</tr>
</tbody>
</table>

Specify the authentication mechanism you want to use when you create a new WS-security profile.

The WS-Security Profiles module lists the WS-security profiles that are currently in effect.

---

### WS-Security Profiles module

Create a new WS-Security profile

Create a new WS Security profile to define how ServiceNow authenticates a Web Services message when WS-Security is enabled.

Role required: web_service_admin or admin

1. Navigate to **System Web Services > WS Security Profiles**.
2. Click **New**.
3. Fill in the WS-Security Profile form (see table).

### WS-Security Profile form fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a unique name for the security profile.</td>
</tr>
<tr>
<td>Type</td>
<td>Select <strong>X509</strong> to verify the request's certificate. Select <strong>Username</strong> to verify the request's user credentials.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Run as user</td>
<td>Select the ServiceNow user the instance will impersonate if authentication succeeds and the <strong>Bind Session</strong> field is selected. All web services requests will be attributed to this user. For example, if you select the <strong>System Administrator</strong> user then the instance treats all web services operations as being done by the system administrator. Make sure the user you select has appropriate SOAP privileges if you are using the <code>glide.soap.strict_security</code> high security setting. This field is only visible when the type is <strong>X509</strong>.</td>
</tr>
<tr>
<td>Order</td>
<td>Enter the order in which the instance checks security profiles. The instance checks all security profiles when processing an incoming SOAP request. If a request fails any security profile authentication requirement, the instance stops processing additional security profiles and fails the request.</td>
</tr>
<tr>
<td>Bind Session</td>
<td>Select this check box to have the instance impersonate the user listed in the <strong>Run as user</strong> field. You should only set this field for one profile at a time. If multiple profiles have this field selected, ServiceNow impersonates the user from the last successfully authenticated WS-Security profile. If no profile has this field selected, ServiceNow impersonates the user provided with the basic authentication headers or impersonates the default user (Guest).</td>
</tr>
<tr>
<td>X509 Certificate</td>
<td>Select the certificate record containing the certificate for web service requests. ServiceNow only validates the request signature. It automatically trusts the certificate's certificate authority (CA). This field is only visible when the type is <strong>X509</strong>.</td>
</tr>
<tr>
<td>Profile action</td>
<td>Select how you want the instance to authenticate the user credentials. Select <strong>Authenticate with user</strong> if you want the instance to authenticate the request based on an existing user record. The request's credentials must match values in an existing user record. Select <strong>Specify user to authenticate</strong> if you want to list a user name and password combination that all web services requests must meet. The request's credentials must match the user name and password you list. This field is only visible when the type is <strong>Username</strong>.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>User field to match UserName</td>
<td>Select the column from the User (sys_user) table containing the value you want match against the request's UserName. For example, if you select Email then the request UserName header must contain an email address matching an existing ServiceNow user. This field is only visible when the profile action is Authenticate with user.</td>
</tr>
<tr>
<td>User name</td>
<td>Enter the user name that all web services requests must contain. This field is only visible when the profile action is Username.</td>
</tr>
<tr>
<td>User password</td>
<td>Enter the password that all web services requests must contain. This field is only visible when the profile action is Username.</td>
</tr>
</tbody>
</table>

4. **Click Submit.**

WSS X.509 Token Profile
Use the X.509 framework for a WSS X.509 security profile.

An X.509 certificate is used to validate a public key that is used to sign the incoming SOAP message. It specifies a binding between a public key and a set of attributes that includes (at least) the following:

- subject name
- issuer name
- serial number
- validity interval

Use the **X.509 authentication framework** as defined by the Web Services Security: SOAP Message Security specification.

Upload the certificate and reference it in the **X509 Certificate** field. If this is a bound session, select the user to impersonate when the WS-Security authentication succeeds.
WSS X.509 Security Profile

WSS UsernameToken Profile
When specifying the X.509 Token Profile, you can also supply a UsernameToken in the SOAP request.

A UsernameToken is used as a means of identifying the requestor by "username", and optionally using a password (or shared secret, or password equivalent) to authenticate that identity to ServiceNow.

There are two ways to authenticate a UsernameToken.
1. Authenticate with existing ServiceNow user credentials.

Authenticate with existing ServiceNow user credentials

Use the username of the incoming SOAP request to look up a user in ServiceNow by the specified User field to match the UserName value. The password value in the incoming UsernameToken is used to authenticate the request. When the Bind session option is selected, the user that authenticates successfully will be used for the session.

2. Authenticate with specified user credentials.
Authenticate with specified user credentials

Authenticate using login credentials unrelated to users in the User table. When the Bind session option is selected, the user that is specified in the Run as user field will be used for the session.

**Note:** The UsernameToken Profile cannot be used independent of the X.509 Token Profile.

Example WS-Security SOAP envelope header

An example of a valid WS-Security SOAP envelope header.

```
<SOAP-ENV:Header>
  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
                SOAP-ENV:mustUnderstand="1">
                               EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
                               ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
                               wsu:Id="CertId-2D914AB929A6719E7F13068829874641"/>
    <wsu:Id="CertId-2D914AB929A6719E7F13068829874641"/>
  </wsse:Security>
</SOAP-ENV:Header>
```
© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
**SOAP security policies**

The Enhanced Web Service Provider - Common plugin adds the SOAP Security Policies module to the System Web Services application.

This module allows administrators to set the following security policies.

- Enable or disable signing SOAP requests when consuming an external web service
- Specify the authentication requirements SOAP requests must meet when communicating over WS-Security.
Administrators can activate the Enhanced Web Service Provider - Common plugin to enable unsigned WS-Security requests and specify what authentication requirements SOAP requests have.

Role required: admin

For evaluation, you can activate the plugin for an application that requires a purchased subscription on a non-production instance. To activate the plugin on production instances, you must purchase the subscription. To purchase a subscription, contact your ServiceNow account manager. For details on purchasing a plugin, see Purchase a plugin.

Some plugins require activation by ServiceNow personnel. Request these plugins through the HI Customer Service System instead of activating them yourself. For details, see Request a plugin.

For plugins that you can activate yourself, continue with the following steps.

1. Navigate to System Definition > Plugins.
2. Find and click the plugin name.
3. On the System Plugin form, review the plugin details and then click the Activate/Upgrade related link.

   If the plugin depends on other plugins, these plugins are listed along with their activation status.

   If the plugin has optional features that depend on other plugins, those plugins are listed under Some files will not be loaded because these plugins are inactive. The optional features are not installed until the listed plugins are installed (before or after the installation of the current plugin).

4. Optional: If available, select the Load demo data check box.

   Some plugins include demo data—Sample records that are designed to illustrate plugin features for common use cases. Loading demo data is a good practice when you first activate the plugin on a development or test instance.

   You can also load demo data after the plugin is activated by clicking the Load Demo Data Only related link on the System Plugin form.

5. Click Activate.

Installed with the Enhanced Web Service Provider - Common plugin

The following components installed with the Enhanced Web Service Provider - Common plugin.

The Enhanced Web Service Provider - Common plugin installs the following components:

<table>
<thead>
<tr>
<th>Component Type</th>
<th>Component Installed</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>Web Services Security Profiles</td>
<td>The plugin adds this module to the System Web Services application.</td>
</tr>
<tr>
<td>System Property</td>
<td>glide.soap.default_security_policy</td>
<td>Specifies the default security policy to use when enforcing Web Services-Security (WSS) for inbound requests.</td>
</tr>
</tbody>
</table>

Certificates required for signed SOAP requests

In order to sign SOAP requests for WS-Security communications, ServiceNow requires the following certificates.

- The X.509 certificate from the requester
Create a new security policy

Administrators can specify which security profiles WS-Security communications must meet by creating a new security policy.

Role required: web_service_admin or admin

2. Click New.
3. Fill out the SOAP Security Policy form (see table).

### SOAP Security Policy form fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a unique name for the security policy. Use this name to set the default security policy with the glide.soap.default_security_policy property.</td>
</tr>
<tr>
<td>Type</td>
<td>Select whether the SOAP security policy applies to inbound or outbound traffic.</td>
</tr>
<tr>
<td>Required to Sign SOAP Request</td>
<td>Select this checkbox to require signed SOAP requests. Clear the checkbox to allow unsigned SOAP requests. When enabled, the instance will produce an error for any SOAP request that does not include a valid signature. When disabled, the instance still verifies any signature included with a SOAP request.</td>
</tr>
<tr>
<td>Authenticate</td>
<td>Select if a SOAP request must authenticate against all security profiles or at least one security profile.</td>
</tr>
<tr>
<td>Security Profiles</td>
<td>Select the security profiles you want to apply to this SOAP security policy. You must select at least one security profile.</td>
</tr>
</tbody>
</table>

4. Click Submit.

Specify requirement for signed SOAP requests

Use a SOAP security policy to specify whether the instance requires signed SOAP requests for all inbound SOAP traffic.

Role required: web_service_admin or admin

By default, all inbound SOAP traffic must be signed. Administrators may want to disable this policy and allow unsigned SOAP requests to ServiceNow web services.

2. Select the Default Policy.
3. Clear the Required to Sign SOAP Request check box (selected by default) to disable the requirement.
4. Click Update.

SOAP default security policy

Administrators can specify the SOAP security policy an instance uses with the system property glide.soap.default_security_policy.
The `glide.soap.default_security_policy` system property specifies the name of the SOAP security policy the instance uses when enforcing Web Services-Security (WSS) for inbound requests.

### SOAP default security policy settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Default value</td>
<td>Default Security Policy</td>
</tr>
<tr>
<td>Location</td>
<td>Add to the System Properties (sys_properties) table</td>
</tr>
</tbody>
</table>

Set the SOAP default security policy.

Set the SOAP default security policy.

Role required: web_service_admin or admin

1. Navigate to System Web Services > Properties.
2. In the **Security Policy to enforce if WS-Security is enabled** field, enter the default security policy to use when enforcing WS-security.
3. Click **Save**.

### WS-Security error logging

The `glide.processor.debug.SOAPProcessor` system property allows error messages about WS-security to be displayed in the transaction log.

The system property `glide.processor.debug.SOAPProcessor` enables (true) or disables (false) debugging messages for SOAP processing such as certificate and keystore checks.

### glide.processor.debug.SOAPProcessor fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>true</td>
</tr>
<tr>
<td>Default value</td>
<td>false</td>
</tr>
<tr>
<td>Location</td>
<td>Add to the System Properties (sys_properties) table</td>
</tr>
</tbody>
</table>

### WS-Security error messages

An instances produces one of the following error messages when it encounters an issue with a WS-security SOAP message.

### WS-security error messages

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Security Policy Selected. Select an Inbound policy for Inbound Requests.</td>
<td>The default policy name is set to an outbound policy. Set the default policy name to an inbound security policy.</td>
</tr>
<tr>
<td>Error</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SOAP request not Signed. Policy requires signing.</td>
<td>The SOAP security policy requires signing and the inbound SOAP request is not signed. Either specify a different SOAP security policy or provide the SOAP request with a proper signature.</td>
</tr>
<tr>
<td>No profiles to authenticate.</td>
<td>The selected Security policy either does not have any security profiles (X509 or UserNameToken) or the security profiles are inactive. Verify at least one security profile is active.</td>
</tr>
<tr>
<td>Unable to verify signed request.</td>
<td>The inbound SOAP request contains an invalid signature. The SOAP request changed after being signed.</td>
</tr>
<tr>
<td>Failed to extract principal(s) from request.</td>
<td>The SOAP request has either been tampered or was not well formed. ServiceNow cannot extract credentials to authenticate the request.</td>
</tr>
<tr>
<td>Failed to authenticate WS-security, unknown type.</td>
<td>The SOAP request contains an unsupported security profile. Resend the request with a supported security profile type: X509 or UsernameToken.</td>
</tr>
<tr>
<td>Failed to authenticate WS-security.</td>
<td>ServiceNow failed to authenticate against the provided profile credentials. Verify that the SOAP request is using the proper credentials.</td>
</tr>
</tbody>
</table>

**WS-Security properties**

These properties control the behavior of WS-Security X.509 tokens.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| glide.soap.msg_digest.algorithm                        | Specifies the method digest algorithm. Possible values are SHA-1, SHA-256, and SHA-512.  
  - Type: String  
  - Default value: SHA-1  
  - Location: Add to the System Property (sys_properties) table  |
| glide.soap.signature.algorithmm                         | Specifies the signature algorithm. Possible values are RSA-SHA-1, RSA-SHA-256, and RSA-SHA-512.  
  - Type: String  
  - Default value: RSA-SHA-1  
  - Location: Add to the System Property (sys_properties) table  |
Each possible value for these properties represents a standard WS-Security algorithm.

**Property value URLs**

<table>
<thead>
<tr>
<th>Value</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method digest algorithms</strong></td>
<td></td>
</tr>
<tr>
<td>SHA-1</td>
<td><a href="http://www.w3.org/2000/09/xmlsig#sha1">http://www.w3.org/2000/09/xmlsig#sha1</a></td>
</tr>
<tr>
<td>SHA-256</td>
<td><a href="http://www.w3.org/2001/04/xmlenc#sha256">http://www.w3.org/2001/04/xmlenc#sha256</a></td>
</tr>
<tr>
<td>SHA-512</td>
<td><a href="http://www.w3.org/2001/04/xmlenc#sha512">http://www.w3.org/2001/04/xmlenc#sha512</a></td>
</tr>
<tr>
<td><strong>Signature algorithms</strong></td>
<td></td>
</tr>
<tr>
<td>RSA-SHA-1</td>
<td><a href="http://www.w3.org/2000/09/xmlsig#rsa-sha1">http://www.w3.org/2000/09/xmlsig#rsa-sha1</a></td>
</tr>
<tr>
<td>RSA-SHA-256</td>
<td><a href="http://www.w3.org/2001/04/xmlsig-more#rsa-sha256">http://www.w3.org/2001/04/xmlsig-more#rsa-sha256</a></td>
</tr>
<tr>
<td>RSA-SHA-512</td>
<td><a href="http://www.w3.org/2001/04/xmlsig-more#rsa-sha512">http://www.w3.org/2001/04/xmlsig-more#rsa-sha512</a></td>
</tr>
<tr>
<td><strong>Canonicalization algorithms</strong></td>
<td></td>
</tr>
<tr>
<td>Canonical xml 1.0</td>
<td><a href="http://www.w3.org/TR/2001/REC-xml-c14n-20010315">http://www.w3.org/TR/2001/REC-xml-c14n-20010315</a></td>
</tr>
<tr>
<td>Canonical xml 1.0 with Comments</td>
<td><a href="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments">http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments</a></td>
</tr>
<tr>
<td>Exclusive Canonical xml 1.0</td>
<td><a href="http://www.w3.org/2001/10/xml-exc-c14n#">http://www.w3.org/2001/10/xml-exc-c14n#</a></td>
</tr>
<tr>
<td>Exclusive Canonical xml 1.0 with comments</td>
<td><a href="http://www.w3.org/2001/10/xml-exc-c14n#WithComments">http://www.w3.org/2001/10/xml-exc-c14n#WithComments</a></td>
</tr>
</tbody>
</table>

**Strict security for web services**

Strict security for web services requires that users meet Contextual Security requirements to access instance resources.

By default, basic authentication for web services only determines whether a user is authorized to access the instance with a SOAP connection. Once authorized, any user can access any table published as a web service.

The system property **Enforce strict security on incoming SOAP requests** changes this behavior and requires that users meet Contextual Security requirements to access instance resources from web services.
With this property enabled, only users that have the proper SOAP role and also meet the access control rule conditions for a given table and operation can perform that operation from a SOAP connection.

**Enforce strict security for inbound SOAP**

Strict security for web services requires that users meet Contextual Security requirements to access instance resources.

Role required: admin

**Note:** ServiceNow does not support digital certificates, digital signatures, or other digested token methods in SOAP web service calls.

To enforce strict security for web services connections:

1. Navigate to **System Properties > Web Services**.
2. Select **Yes** for **Enforce strict security on incoming SOAP requests**.

**Mutual authentication for web services**

ServiceNow supports mutual authentication for outbound web services.

**Overriding the SOAP endpoint**

The SOAP endpoint address where the SOAP message is posted is consistent with the endpoint of the WSDL.

In some cases, however, the WSDL may reference an incorrect endpoint URL. If this happens, it is necessary to over-ride the generated URL by creating the property `com.glide.soap_address_base_url` to contain the new URL. You may have to add the property manually as this is not an out-of-box property.

For instance, a generated SOAP endpoint may look like this:

```plaintext
https://instance.service-now.com/incident.do?SOAP
```

You can specify a property to define the endpoint such that it goes through a proxy by setting the property:

```plaintext
com.glide.soap_address_base_url = "https://myproxy.mycompany.com/service-now/
```

This will result in the endpoint being generated to appear as:

```plaintext
https://myproxy.mycompany.com/service-now/incident.do?SOAP
```
Enabling HTTP compression

By default, the SOAP request is accepted un-compressed and the result of the request is returned un-compressed.

To enable HTTP compression using (gzip) when sending in your SOAP request, set the following HTTP header:

```
Content-Encoding: gzip
```

To receive the SOAP response compressed using (gzip) send in your SOAP request with the following HTTP header:

```
Accept-Encoding: gzip
```

Debugging incoming SOAP envelope

To capture incoming SOAP envelope XML in the system log, add the property `glide.processor.debug.SOAPProcessor` with a value of `true`.

When enabled, this property adds the incoming SOAP envelope in the Message field of the system log (System Logs > All). Disable this debugging feature as soon as you are finished so that the log is not overwhelmed with excessive and unnecessary debugging information.

Preventing empty elements in SOAP messages

By default, an instance does not omit empty elements, elements with NULL or NIL values, from SOAP messages.

To prevent SOAP responses from containing empty elements, an administrator can create a system property called `glide.soap.omit_null_values` and set the value to `true`. This behavior is compliant with the WSDL as all elements in a SOAP message have a minOccurs=0 attribute and are therefore optional. In addition, this behavior prevents the instance from creating inefficient SOAP messages containing a large number of empty elements.

Set this property to `false` to allow SOAP messages to search for existing fields with empty values. For example, if an administrator wants to search for incidents with an empty Assigned to field from a SOAP message, then the SOAP message must be able to send an empty value for this field.

Note: Changing the value of this property may cause unintended actions in existing web service integrations. Administrators are strongly encouraged to carefully test the new behavior to ensure that existing integrations process empty elements as expected.

Insert related records using SOAP

Support is available for inserting hierarchical data into tables or web service import set tables. The hierarchical data in the Insert API is automatically mapped to related records of the targeted table.

Role required: admin

Create and set the property `glide.web_service.hierarchical` to `true`.

The client of the API can also override this value by invoking the SOAP web service with the URL parameter `hierarchical=true`. 
For example, when a related list is created for the incident table called `u_custom_comments`:

Hierarchical Incident

And `u_comment_items` is created as a related list for `u_custom_comments`:
Hierarchical Custom Comments

WSDL Schema with related records

When a WSDL for the target Incident table is requested with an additional parameter of `hierarchical=true`, the WSDL schema for the Insert function will reflect available related records that may participate in the hierarchical data payload.

For example, the insert portion of the schema of the incident table, when requested with `hierarchical=true` displays its hierarchy as follows:

https://instance.service-now.com/incident.do?WSDL&hierarchical=true
<xsd:element name="insert">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="0" name="active" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="activity_due" type="xsd:string"/>
      <xsd:element minOccurs="0" name="approval" type="xsd:string"/>
      <xsd:element minOccurs="0" name="approval_history" type="xsd:string"/>
      <xsd:element minOccurs="0" name="approval_set" type="xsd:string"/>
      <xsd:element minOccurs="0" name="assigned_to" type="xsd:string"/>
      <xsd:element minOccurs="0" name="assignment_group" type="xsd:string"/>
      <xsd:element minOccurs="0" name="business_duration" type="xsd:string"/>
      <xsd:element minOccurs="0" name="business_stc" type="xsd:integer"/>
      <xsd:element minOccurs="0" name="calendar_duration" type="xsd:string"/>
      <xsd:element minOccurs="0" name="calendar_stc" type="xsd:integer"/>
      <xsd:element minOccurs="0" name="caller_id" type="xsd:string"/>
      <xsd:element minOccurs="0" name="category" type="xsd:string"/>
      <xsd:element minOccurs="0" name="closed" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="create_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="create_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="creator" type="xsd:string"/>
      <xsd:element minOccurs="0" name="directory" type="xsd:string"/>
      <xsd:element minOccurs="0" name="entity" type="xsd:string"/>
      <xsd:element minOccurs="0" name="entity_unique_key" type="xsd:string"/>
      <xsd:element minOccurs="0" name="execute_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="execute_status" type="xsd:string"/>
      <xsd:element minOccurs="0" name="execute_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="expiration_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="expiration_type" type="xsd:string"/>
      <xsd:element minOccurs="0" name="inventory_name" type="xsd:string"/>
      <xsd:element minOccurs="0" name="inventory_type" type="xsd:string"/>
      <xsd:element minOccurs="0" name="is_alert" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="is_cancelable" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="is_completed" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="is_canceled" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="is_expired" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="is_escalated" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="is_flagged" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="is_recurrence" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="is_renewable" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="is_rescheduled" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="is_uom" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="is_valid" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="last_activity" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_activity_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_activity_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_canceled" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="last_canceled_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_canceled_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_completed" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="last_completed_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_completed_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_create" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="last_create_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_create_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_escalated" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="last_escalated_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_escalated_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_flagged" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="last_flagged_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_flagged_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_rescheduled" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="last_rescheduled_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_rescheduled_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_uom" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="last_valid" type="xsd:boolean"/>
      <xsd:element minOccurs="0" name="last_valid_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_valid_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_workflow_activity" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_workflow_activity_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="last_workflow_activity_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="name" type="xsd:string"/>
      <xsd:element minOccurs="0" name="note" type="xsd:string"/>
      <xsd:element minOccurs="0" name="notes" type="xsd:string"/>
      <xsd:element minOccurs="0" name="number" type="xsd:string"/>
      <xsd:element minOccurs="0" name="organization" type="xsd:string"/>
      <xsd:element minOccurs="0" name="oe_number" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parent" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parent_entity" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parent_entity_unique_key" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parent_type" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parent_workflow_activity" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parent_workflow_activity_date" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parent_workflow_activity_user" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parent_workflow_state" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowActivity" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowState" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStateId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatus" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusName" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusName" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusName" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusName" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusName" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusName" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusName" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusName" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusName" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusName" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusName" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusNameId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="parentWorkflowStatusId" type="xsd:string"/>
WSDL Schema Continued

The WSDL above shows the incident table having a related table `u_custom_comments` that itself has a related table `u_comment_items`.

Hierarchical SOAP Message

When the SOAP message is constructed from the hierarchical web service described by the WSDL and invoked, it will create the `incident`, `u_custom_comments`, and `u_comment_items` records.

Endpoint URL

https://instance.service-now.com/incident.do?SOAP&hierarchical=true
Request

```xml
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:inc="http://www.service-now.com/incident">
  <soapenv:Header/>
  <soapenv:Body>
    <inc:insert>
      <short_description>test hierarchical</short_description>
      <u_custom_comments>
        <u_comment>comment 1</u_comment>
        <u_comment_type>travel</u_comment_type>
        <u_comment_items>
          <u_name>name 1</u_name>
          <u_value>value 1</u_value>
        </u_comment_items>
      </u_custom_comments>
    </inc:insert>
  </soapenv:Body>
</soapenv:Envelope>
```

Response

```xml
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:inc="http://www.service-now.com/incident">
  <soapenv:Header/>
  <soapenv:Body>
    <insertResponse>
      <sys_id>8422ebe7c0a8006e7d23848c2dc8ba47</sys_id>
      <number>INC0010001</number>
    </insertResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP session management and reporting

A SOAP session is a Glide session established with an instance by any external SOAP client, such as a web services client application, a ServiceNow MID Server, or the ServiceNow ODBC driver.

SOAP sessions are included in the list of user sessions at User Administration > Logged in users. SOAP sessions are identified by the ?SOAP URLs.

SOAP session properties

Certain properties control how SOAP sessions are maintained.
SOAP session properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>glide.soap.invalidate_session_timeout</td>
<td>The duration, in seconds, that an active session will remain open. After this duration is reached, the instance deactivates the session and reclaims any system resources. If the client sends another request after the timeout duration is reached, the instance will establish a new session. This property accepts values between 5 seconds and 1200 seconds (20 minutes).</td>
</tr>
<tr>
<td></td>
<td>· Type: integer</td>
</tr>
<tr>
<td></td>
<td>· Default value: 60</td>
</tr>
<tr>
<td></td>
<td>· Location: Add to the System Properties [sys_properties] table</td>
</tr>
</tbody>
</table>

View a SOAP session log

You can view a user's log from a SOAP session.

Role required: admin

1. Navigate to User Administration > Logged in users.

2. Open an active SOAP session to see the transactions log.
   The SOAP session is marked as inactive within 60 seconds of the last transaction.

Long-running SOAP request support

The Now Platform supports long-running SOAP requests by preventing socket timeouts due to inactivity of the network connection while the requests are in process.

This functionality improves the efficiency of the ODBC driver when requesting large numbers of records, doing aggregate queries, or using order by expressions that require sorting.
By default, ServiceNow provides timeout protection for web services clients provided by ServiceNow such as the ODBC driver and the MID Server. You can add time timeout protection to customer developed web services with system properties.

### Timeout Protection

Web services clients receive a 307-Temporary Redirect to keep long sessions alive and prevent a timeout due to socket inactivity. A 307-Temporary Redirect causes web services clients which support the status code to repeat their last request to the location specified in the HTTP location header. The value of the location header sent by ServiceNow is the same URL that the web services client originally specified. The use of 307-Temporary Redirects is WS-I compliant.

A web service request that exceeds the timeout limit (`glide.soap.request_processing_timeout`) can only receive a 307-Temporary Redirect when all of these conditions are met:

- The value of `glide.soapprocessor.allow_long_running_threads` is true.
- The request includes a `redirectSupported=true` URL parameter.
- The request is session-aware (supports HTTP cookies).
- The number of redirects has not exceeded the value set by `glide.soap.max_redirects`.

If any of these conditions is not met, the web service client receives a 408 Request Timeout error.

**Note:** To ensure that applications experience a socket timeout rather than a 408 Request Timeout, set the `glide.soap.request_processing_timeout` property to a value larger than the shortest socket timeout setting in effect for the connection between the application and the ServiceNow instance (300 seconds for hosted instances).

### Long-running SOAP request properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>glide.http.connection_timeout</td>
<td>Specify the maximum number of milliseconds an outbound HTTP request (such as Web Services) will wait to establish a connection.</td>
</tr>
<tr>
<td></td>
<td>- Type: integer</td>
</tr>
<tr>
<td></td>
<td>- Default value: 10000 (10 seconds)</td>
</tr>
<tr>
<td></td>
<td>- Location: system properties (<code>sys_properties</code>) table</td>
</tr>
<tr>
<td>glide.http.timeout</td>
<td>Specifies the maximum number of milliseconds to wait before an outbound transaction times out.</td>
</tr>
<tr>
<td></td>
<td>- Type: integer</td>
</tr>
<tr>
<td></td>
<td>- Default value: 175000 (175 seconds)</td>
</tr>
<tr>
<td></td>
<td>- Location: Add to system properties (<code>sys_properties</code>) table</td>
</tr>
<tr>
<td>glide.soap.max_redirects</td>
<td>Specifies the maximum number of redirects the server sends to the client before the soap request is timed out.</td>
</tr>
<tr>
<td></td>
<td>- Type: integer</td>
</tr>
<tr>
<td></td>
<td>- Default value: 20</td>
</tr>
<tr>
<td></td>
<td>- Location: system properties (<code>sys_properties</code>) table</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>glide.soap.request_processing_timeout</td>
<td>Specify the maximum number of seconds an inbound SOAP request has to finish processing before the connection times out. This property computes a default value from the value of the property glide.http.timeout divided by 1000. This property accepts values 5–1200 seconds (20 minutes). Customers might have network infrastructure (such as proxy servers) in place which implement a shorter timeout. In this case, a socket timeout may occur unless this property is set to a shorter value. Set this property to a value several seconds less than the shortest socket inactivity timeout in effect anywhere in the network path between the client application and the instance.</td>
</tr>
<tr>
<td>glide.soapprocessor.allow_long_running_threads</td>
<td>Enables or disables a 307-Temporary Redirect when the request includes a redirectSupported=true parameter. The default setting is true (enabled).</td>
</tr>
<tr>
<td>glide.soapprocessor.max_long_running_threads</td>
<td>Controls the maximum number of long-running SOAP threads which can run at any one time. The default value for this property is determined dynamically based on the number of SOAP semaphores configured. It should not be necessary to change this value.</td>
</tr>
</tbody>
</table>

**AttachmentCreator SOAP web service**

Attach documents to records in ServiceNow by sending a SOAP message targeting the ecc_queue table.

**Important:** The AttachmentCreator SOAP web service is not recommended. Instead, use the REST Attachment API to manage attachments with web services.

Using the AttachmentCreator SOAP web service, you can attach a single document to a message that is a maximum of 5 MB. The following is an example of a URL or target end point: https://instance_name.service-now.com/ecc_queue.do?WSDL

**ecc_queue Fields for Attachment Creation**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent</td>
<td>The name of the agent sending in the request, this can be any value since it is not used for processing.</td>
<td>AttachmentCreator</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>topic</strong></td>
<td>The topic of the queue record, this value must be set to &quot;AttachmentCreator&quot; to trigger the attachment creation</td>
<td>AttachmentCreator</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td>This field must contain a &quot;:&quot; delimited value of the file name, and its content-type</td>
<td>file_name.xls:application/vnd.ms-excel</td>
</tr>
<tr>
<td><strong>source</strong></td>
<td>This field must contain a &quot;:&quot; delimited value of the target table and its sys_id</td>
<td>incident:dd90c5d70a0a0b39000aac5aaee704ae8</td>
</tr>
<tr>
<td><strong>payload</strong></td>
<td>This field must contain the Base64 encoded string representing the object to be attached</td>
<td>the base 64 encoded string</td>
</tr>
</tbody>
</table>

Sending in the values described in the table above will attach an Excel file to the incident table for the record located by the sys_id dd90c5d70a0a0b39000aac5aaee704ae8.

**Security**

Like all other HTTP based web services available on the platform, the AttachmentCreator SOAP web service is required to authenticate using basic authentication by default. The user ID that is used for authentication will be subjected to access control in the same way as an interactive user.

To create attachments, the SOAP user must have any roles required to create Attachment (sys_attachment) records, as well as the soap_create role, and any roles required to read and write records on the target table, such as the itil role to add attachments to incident records. By default there is no single role allowing you to add attachments. You can create a role to explicitly allow adding attachments, then assign this role to the SOAP user.

**File type security**

You can control what file types users can attach by setting the `glide.attachment.extensions` and `glide.security.file.mime_type.validation` properties.

For these properties to apply to the AttachmentCreator web service, the property `glide.attachment.enforce_security_validation` must be set to true. This property is true by default.

**Example SOAP Message**

The following is an example of a SOAP message that would take a text file 'john1.txt' of mime-type: text/plain and attach it to an Incident with a GUID of: e6eed6950a0a3c59006f32c8e3ff3cf9. Note the payload is the base64 encoding of the file itself.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:ecc="http://www.service-now.com/ecc_queue">
  <soapenv:Header />
  <soapenv:Body>
    <ecc:insert>
      <agent>AttachmentCreator</agent>
    </ecc:insert>
  </soapenv:Body>
</soapenv:Envelope>
```
Example Node.js Script

The following example Node.js script adds an attachment to an incident record. Run this script from a client computer, not an instance.

```javascript
/**
 * Node.js to ServiceNow attachment upload via SOAP
 * Andrew Venables andrew.venables@servicenow.com
 * July 2014
 * Version 1.0
 */

var soap = require('soap'), // https://github.com/vpulim/node-soap
    mime = require('mime'), // https://github.com/broofa/node-mime
    fs = require('fs');

var WSDL_FILENAME = 'ecc_queue.xml'; // Goto https://instancename.service-now.com/ecc_queue.do?WSDL and save a copy of the WSDL locally for simplicity
var DIRECTORY_CONTAINING_FILES = '/Users/andrew.venables/Documents/Uploads'; // Local path to the directory containing all the files we want to upload
var USERNAME = 'andy.venables'; // An admin user account on the instance
var PASSWORD = 'MY_PASSWORD'; // Password for above account
var TARGET_TABLE = 'incident'; // Target table to attach the files to
var TARGET_SYS_ID = '9d385017c611228701d22104cc95c371'; // Target record / sys_id to attach the files to. OOTB INC00000002

var files_to_upload; // Global that will contain our list of files to be uploaded
var pos = 0; // Global pointer for our position in the files_to_upload list

// Create a SOAP client to use to post to the instance
soap.createClient(WSDL_FILENAME, function(err, client) { // Node uses callbacks
    if (err) console.error(err);

    // Set the username and password
    client.setSecurity(new soap.BasicAuthSecurity(USERNAME, PASSWORD));

    // Read all the files in our source directory, will include . and ..
    files_to_upload = fs.readdirSync(DIRECTORY_CONTAINING_FILES);

    console.log('Files to upload: ' + files_to_upload.length + '\n');

    // Start iterating through the list of files to upload
    next(client);
});
```
// Process the next file in the files_to_upload array
// This is a neat way of dealing with Node and its expectation of callbacks
function next(client) {
    // Check we haven't reached the end
    if (pos >= files_to_upload.length) return;

    // Get the next file to upload
    var file_name = files_to_upload[pos];

    // Increment the pointer to the next file
    pos++;

    console.log(pos + '/' + files_to_upload.length + ' - Uploading file: ' + file_name);

    // A blank file is the end of the list
    if (file_name == '') return;

    // Skip to the next file as this one is invalid
    if (file_name == '.' || file_name == '..' || file_name.indexOf('.') == 0)
        next(client);

    // Get the file type using an module called mime
    var file_type = mime.lookup(file_name);
    console.log('   of type: ' + file_type);

    var file_payload;

    // Load the file into a buffer
    fs.readFile(DIRECTORY_CONTAINING_FILES + '/' + file_name, function(err, the_data) {
        if (err) console.error(err);

        // Encode the buffer to base64
        file_payload = new Buffer(the_data, 'binary').toString('base64');

        // Set the parameters before we call the Web Service
        var parameters = {
            'agent': 'AttachmentCreator',
            'topic': 'AttachmentCreator',
            'name': file_name + ':' + file_type,
            'source': TARGET_TABLE + ':' + TARGET_SYS_ID,
            'payload': file_payload
        };

        console.log('      sending...')
        // Make the Web Service call, remember node likes callbacks
        client.insert(parameters, function(err, result) {
            if (err) console.error(err);

            console.log(result);

            // This file is done, next!
            next(client);
        });
    });
}
Example Perl Script

The following example Perl script will create an attachment to an incident record.

```perl
use MIME::Base64;
use strict;
use SOAP::Lite;

# the ServiceNow instance
my $SNC_HOST = "https://instance_name.service-now.com";
my $base64;
my $buf;

# upload and attach a file on the local disk, base 64 encode it into a string first
open(FILE, "/Users/davidloo/Desktop/test_files/number_test.xls") or die "!";
binmode FILE; # preserves file formatting on Windows
while (read(FILE, $buf, 60*57)) {
    $base64 .= encode_base64($buf);
}

# call the sub routine to attach
attach_incident();

sub attach_incident {
    # target the ecc_queue table
    my $soap = SOAP::Lite->proxy("$SNC_HOST/ecc_queue.do?SOAP");
    $soap->{_transport}->{_proxy}->{ssl_opts}->{verify_hostname} = 0;
    my $method = SOAP::Data->name('insert')->attr({xmlns => 'http://www.service-now.com/' });

    # set the ecc_queue parameters
    my @params = (SOAP::Data->name(agent => 'AttachmentCreator'));
    push(@params, SOAP::Data->name(topic => 'AttachmentCreator'));

    # identify the file name and its mime type
    push(@params, SOAP::Data->name(name => 'number_test.xls:application/vnd.ms-excel'));

    # attach to the incident, specifying its sys_id
    push(@params, SOAP::Data->name(source => 'incident:dd90c5d70a0a0b39000aac5ae704ae8'));

    # set the payload to be the base 64 encoded string representation of the file
    push(@params, SOAP::Data->name(payload => $base64));

    # invoke the web service
    my $result = $soap->call($method => @params);
    print_fault($result);
    print_result($result);
}

sub print_result {
    my ($result) = @_;
    if ($result->{body} && $result->{body}->{'insertResponse'}) {
        my %keyHash = %{ $result->{body}->{'insertResponse'} };
        foreach my $k (keys %keyHash) {
            print
        }
    } else { ...
}
```

Perl API

The Perl API provides a library of Perl classes and sub routines for programmatic access to the platform and its applications. The API utilizes the SOAP web service interface of the platform. For more information on the Perl programming language, see [www.perl.org](http://www.perl.org).

Perl API system requirements

This Perl API requires a minimum Perl version and several modules.

The ServiceNow Perl API requires Perl 5.8 (or later) with the following modules installed:

- SOAP::Lite (prerequisites [http://soaplite.com/prereqs.html](http://soaplite.com/prereqs.html)) 0.71 or later
- Crypt::SSLeay
- IO::Socket::SSL
- File::Basename
- MIME::Types
- MIME::Type
- MIME::Base64

**Attention:** Certain versions of the Perl API may not be compatible with the ServiceNow software and may cause 443 connection errors. ServiceNow recommends against using versions 5.14.2, 5.13.6, and 5.16.1 of Perl.

Install the Perl API

Download and install the Perl API.

1. Download the Perl API.

   The Perl API is available for use under the terms defined in the Open Source Initiative OSI - Apache License, Version 2.0 license agreement.

   Apache License, Version 2.0

   Apache License
   Version 2.0, January 2004
   http://www.apache.org/licenses/
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.


Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License. You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional
terms or conditions. Notwithstanding the above, nothing herein shall supersedes or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

2. Unpack the archive into a directory.
3. Run the installation commands.

Note: Make sure you are a privileged user on the system.

perl Makefile.PL
make
make test
make install
After installing, you can find documentation for this module with the perldoc command.

```
perldoc ServiceNow
```

**Perl API architecture**

This reference explains the Perl API architecture.
Perl API overview

- ServiceNow.pm
  - createIncident()
  - queryTicket()
  - updateTask()
  - ...

- Incident.pm
  - insert()
  - query()
  - update()
  - ...

- Task.pm
  - insert()
  - query()
  - update()
  - ...

- GlideRecord.pm
  - insert()
  - query()
  - update()
  - ...

Connection.pm
Configuration.pm
ServiceNow.pm

At the top most of the class hierarchy is the ServiceNow.pm module. This module provides direct subroutines that delegate down into subroutines of the object heirarchy, providing a convenient calling convention if object oriented Perl programming is not a desired practice.

ITIL Objects

The next layer of the class hierarchy contains the ITIL Objects for example Incident.pm, and Change.pm. These objects extend the Task.pm object which inherits from the GlideRecord.pm object.

Configuration.pm

To use the Perl API, the programmer instantiates a Configuration.pm object and sets the service endpoint as well as the login credentials to use the API. For example:

```perl
my $CONFIG = ServiceNow::Configuration->new();
$CONFIG->setSoapEndpoint("https://instance_name.service-now.com/");
my $incident = ServiceNow::ITIL::Incident->new($CONFIG);
```

Connection.pm

The Connection.pm class defines the interfaces for making the SOAP calls to the platform. The default implementation uses the SOAP::Lite module, if you wish to implement your own SOAP interface, you can override this class with your own and implement the following subroutines:

- new
- open
- send
- close

If running behind a firewall, edit Connection.pm to specify information about the HTTP proxy. In Connection.pm, change the line:

```perl
$me->{'SOAP'} = SOAP::Lite->proxy($CONFIG->getSoapEndpoint($target));
```

to:

```perl
$me->{'SOAP'} = SOAP::Lite->proxy($CONFIG->getSoapEndpoint($target), proxy => ['https' => 'http://myproxy.mycompany.com/']);
```

Substitute the appropriate proxy URL.

Perl API classes

The Perl API includes multiple classes.
### API

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceNow.pm</td>
<td>A class that contains data querying, creation, and modification subroutines for all the ITIL objects. It is a convenient entry point to the API without using the object oriented ITIL objects directly</td>
</tr>
</tbody>
</table>

### ITIL objects

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incident.pm</td>
<td>The incident class, extends Task, which extends GlideRecord. Maps to the incident table.</td>
</tr>
<tr>
<td>Problem.pm</td>
<td>The problem class, extends Task, which extends GlideRecord. Maps to the problem table.</td>
</tr>
<tr>
<td>Change.pm</td>
<td>The change request class, extends Task, which extends GlideRecord. Maps to the change_request table.</td>
</tr>
<tr>
<td>Request.pm</td>
<td>The service request class, extends Task, which extends GlideRecord. Maps to the sc_request table.</td>
</tr>
<tr>
<td>RequestedItem.pm</td>
<td>The service request item class, extends Task, which extends GlideRecord. Maps to the sc_req_item table.</td>
</tr>
<tr>
<td>SC_Task.pm</td>
<td>The service request task class, extends Task, which extends GlideRecord. Maps to the sc_task table.</td>
</tr>
<tr>
<td>Task.pm</td>
<td>The task class extends GlideRecord. Maps to the task table.</td>
</tr>
<tr>
<td>Ticket.pm</td>
<td>The ticket class, extends Task, which extends GlideRecord. Maps to the ticket table.</td>
</tr>
</tbody>
</table>

### Other

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration.pm</td>
<td>The API configuration class, this class must be passed to the constructor of the other API classes. It defines the SOAP endpoint as well as the credentials to use for API access.</td>
</tr>
<tr>
<td>GlideRecord.pm</td>
<td>The class behind all table access classes.</td>
</tr>
<tr>
<td>Connection.pm</td>
<td>The Connection class implements the web service access layer in an interface that can be overridden. By default, It uses the SOAP::Lite package for web services support.</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Approval.pm</td>
<td>The approval class extends GlideRecord. Maps to the <code>sysapproval_approver</code> table.</td>
</tr>
<tr>
<td>Dictionary.pm</td>
<td>The dictionary class extends GlideRecord. Maps to the <code>sys_dictionary</code> table.</td>
</tr>
<tr>
<td>Attachment.pm</td>
<td>The attachment class extends GlideRecord. Used to add attachments to the other objects.</td>
</tr>
</tbody>
</table>

**Perl API ServiceNow**

The ServiceNow perl module is a collection of Perl subroutines that provides convenient and direct access to the Now Platform.

**ServiceNow Perl API constructor**

This page lists the constructor for the ServiceNow Perl API.

```perl
new(Configuration object, optional Instance ID)
```

Example:

```perl
$config = ServiceNow::Configuration->new();
$SN = ServiceNow->new($config);
```

Sets up an instance of the ServiceNow object using a Configuration object and an optional Instance ID.

**ServiceNow Perl API subroutines**

The ServiceNow Perl API includes multiple subroutines.

### Incident

<table>
<thead>
<tr>
<th>createIncident</th>
<th>createIncident(optional parameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>$number = $SN-&gt;createIncident({&quot;short_description&quot; =&gt; &quot;this is the short description&quot;});</td>
<td>Create an incident. Returns an incident number upon success. On failure returns undef.</td>
</tr>
</tbody>
</table>

```perl
queryIncident(Reference to named parameters hash of Incident fields and exact values)
```

Example:

```perl
@results = $SN->queryIncident({'number' => 'INC0000054'});
```

Query for Incidents matching specified criteria. Returns an array of incident records.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>updateIncident</code></td>
<td><code>updateIncident($number, optional hash of Incident fields and values to update)</code></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>```perl</td>
</tr>
<tr>
<td></td>
<td>$ret = $SN- &gt;updateIncident($number, {$field =&gt; $value});</td>
</tr>
<tr>
<td></td>
<td>Update a ServiceNow incident                                                          Returns undef on failure, all other values indicate success.</td>
</tr>
<tr>
<td><code>closeIncident</code></td>
<td><code>closeIncident(incident number, optional hash of Incident fields and values to update)</code></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>```perl</td>
</tr>
<tr>
<td></td>
<td>$number = $SN- &gt;closeIncident ($number, {$field =&gt; $value});</td>
</tr>
<tr>
<td></td>
<td>Close a ServiceNow incident and optionally update field values. Returns the incident number on success, undef on failure.</td>
</tr>
<tr>
<td><code>reopenIncident</code></td>
<td><code>reopenIncident(incident number)</code></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>```perl</td>
</tr>
<tr>
<td></td>
<td>my $ret = $SN- &gt;reopenIncident('INC99999');</td>
</tr>
<tr>
<td></td>
<td>Reopen a closed ServiceNow incident. Returns the incident number on success, undef on failure.</td>
</tr>
<tr>
<td><code>reassignIncident</code></td>
<td><code>reassignIncident(Incident number, assignment group, optional assigned to)</code></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>```perl</td>
</tr>
<tr>
<td></td>
<td>my $ret = $SN- &gt;reassignIncident('INC99999', 'SOME_GROUP');</td>
</tr>
<tr>
<td></td>
<td>my $ret = $SN- &gt;reassignIncident('INC99999', 'SOME_GROUP', 'username');</td>
</tr>
<tr>
<td></td>
<td>Reassign a ServiceNow Incident to a new assignment_group.(and optionally specify an assigned_to) Returns the incident number on success, undef on failure.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>createTicket</code></td>
<td>Create a ServiceNow ticket associated with an incident. Returns a ticket number upon success.</td>
</tr>
<tr>
<td><code>updateTicket</code></td>
<td>Updates a ServiceNow ticket with specified fields and values. Returns undef on failure.</td>
</tr>
<tr>
<td><code>queryTicket</code></td>
<td>Query for tickets matching specified criteria. Returns undef if no records found.</td>
</tr>
<tr>
<td><code>closeTicket</code></td>
<td>Close a ServiceNow ticket and optionally specify work effort. Returns the ticket number.</td>
</tr>
</tbody>
</table>

Example:

```perl
def createTicket(Reference to named parameters hash of ticket fields and values):
    my $number = $SN-
    >createTicket("
      "category" =>
      "hardware"));

def updateTicket(The ticket number , Reference to named parameters hash of ticket fields and values to update):
    my $ret = $SN-
    >updateTicket($number, {$field => $value});

def queryTicket(Reference to named parameters hash of Incident fields and exact values):
    my @tickets = $SN-
    >queryTicket({'number' => $number});

def closeTicket(Ticket number, {$field => $value})
    my $ret = $SN-
    >closeTicket($number);
    my $ret = $SN-
    >closeTicket($number, {"comments" => "ticket closed"});
```

Create a ServiceNow ticket associated with an incident. Returns a ticket number upon success. On failure undef.

Returns undef on failure, all other values indicate success.

Query for tickets matching specified criteria. Reference to array of hashes of all matching tickets, undef on failure or if no records found.

Close a ServiceNow ticket and optionally specify work effort. Returns the ticket number on success, undef on failure.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reopenTicket</td>
<td>Reopen a closed ServiceNow ticket. Returns the ticket number on success, undef on failure.</td>
</tr>
<tr>
<td>reassignTicket</td>
<td>Reassign a ServiceNow ticket to a new assignment_group. Returns the ticket number on success, undef on failure.</td>
</tr>
<tr>
<td>createRequestedItem</td>
<td>Create a Service Catalog RequestedItem (and indirectly the associated Request and Tasks). Returns a RequestedItem number upon success. On failure returns undef.</td>
</tr>
</tbody>
</table>
| **queryRequestedItem** | **Example:**  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>queryRequestedItem(Reference to named parameters hash of RequestedItem fields and exact values)</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>my $requestedItems = $SN&gt;queryRequestedItem({'number' =&gt; 'SOME_RI_NUMBER'});</td>
<td></td>
</tr>
<tr>
<td>Query for RequestedItems matching specified criteria. Reference to array of hashes of all matching RequestedItem, undef on failure or if no records found.</td>
<td></td>
</tr>
</tbody>
</table>

| **queryRequest** | **Example:**  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>queryRequest(Reference to named parameters hash of Request fields and exact values)</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>my $requests = $SN&gt;queryRequest({'number' =&gt; 'SOME_REQUEST_NUMBER'});</td>
<td></td>
</tr>
<tr>
<td>Query for Requests matching specified criteria. Reference to array of hashes of all matching Request, undef on failure or if no records found.</td>
<td></td>
</tr>
</tbody>
</table>

## Task

| **createTask** | **Example:**  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>createTask(optional parameters)</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>$number = $SN&gt;createTask({&quot;short_description&quot; =&gt; &quot;this is the short description&quot;});</td>
<td></td>
</tr>
<tr>
<td>Create a task record. Returns an task number upon success. On failure returns undef.</td>
<td></td>
</tr>
</tbody>
</table>

| **closeTask** | **Example:**  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>closeTask(task number, optional work effort in seconds)</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>my $ret = $SN-&gt;closeTask($number); my $ret = $SN-&gt;closeTask($number, {$field =&gt; $value});</td>
<td></td>
</tr>
<tr>
<td>Close a ServiceNow Task and optionally update field values. Returns true on success, undef on failure.</td>
<td></td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>reopenTask</td>
<td>Reopen a closed ServiceNow Task. Returns true on success, undef on failure.</td>
<td><code>my $ret = $SN-&gt;reopenTask('TASK99999');</code></td>
</tr>
<tr>
<td>reassignTask</td>
<td>Reassign a ServiceNow Task to a new assignment_group. (and optionally specify an assigned_to) Returns true on success, undef on failure.</td>
<td><code>my $ret = $SN-&gt;reassignTask('TASK99999', 'SOME_GROUP');</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>my $ret = $SN-&gt;reassignTask('TASK99999', 'SOME_GROUP', 'username');</code></td>
</tr>
<tr>
<td>updateTask</td>
<td>Update a ServiceNow Task. Returns true on success, undef on failure.</td>
<td><code>my $ret = $SN-&gt;updateTask($number, {$field =&gt; $value});</code></td>
</tr>
</tbody>
</table>
### queryTask

queryTask(reference to named parameters hash of Task fields and exact values)

Example:

```perl
my @tasks = $SN->queryTask({'number' => $number});
foreach my $task (@tasks) {
    print "Incident number: $task->{'number'}\n";
    print "Assignent Group: $task->{'assignment_group'}\n";
    print "Opened by:       $task->{'opened_by'}\n";
    print "SD:       $task->{'short_description'}\n";
    print "TW:       $task->{'time_worked'}\n";
}
```

Query for Tasks matching specified criteria. Array of hashes of all matching Tasks, undef on failure or if no records found.

### Journal

#### queryJournal

queryJournal(Incident/Ticket/Task number, optional journal field name)

Query for journals entries for the specified incident, ticket or task. Return array of hashes of all matching Journals, undef on failure or if no records found. There will be one hash per per journal entry, 'value' will contain the journal entry string, 'element' will be the name of the field (e.g. 'comments', 'work_notes', etc.)

#### appendJournal

appendJournal(Incident/Ticket/Task number, field name, journal text)

Example:

```perl
my $ret = $SN->appendJournal('INC99999', 'comments', "some comment text");
```

Append a journal entry to the specified journal field of an incident, ticket, or task. Returns true on success, undef on failure.
### Approval

<table>
<thead>
<tr>
<th>queryApproval</th>
<th>queryApproval(reference to named parameters hash of approval fields and exact values)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>my @approvals = $SN-&gt;queryApproval({'approver' =&gt; 'username'});</td>
</tr>
<tr>
<td></td>
<td>Query for approvals Return array of hashes of all matching approvals, undef on failure or if no records found.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>approve</th>
<th>approve(sys_id of the approval, approval state, optional comment text)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>my $ret = $SN-&gt;approve($sys_id, 'Approved');</td>
</tr>
<tr>
<td></td>
<td>my $ret = $SN-&gt;approve($sys_id, 'Rejected', &quot;Please do something else&quot;);</td>
</tr>
<tr>
<td></td>
<td>Approve/reject a ServiceNow approval request and optionally provide a comment. Returns the sys_id on success, undef on failure.</td>
</tr>
</tbody>
</table>

### Dictionary

<table>
<thead>
<tr>
<th>queryFields</th>
<th>queryFields(table, optional boolean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>my @fields = $SN-&gt;queryFields('incident');</td>
</tr>
<tr>
<td></td>
<td>List all the fields of an Incident, Request, RequestedItem or Task. Returns a reference to a hash of fields in the specified table type. The hash key is the field name, and the hash value is a hash reference to attributes about the field: ‘mandatory’, ‘hint’, ‘label’, ‘reference’ and ‘choice’. Returns undef on failure. If getchoices is true then ‘choices’ is a reference to a hash containing individual choices, keyed by choice value and containing choice ‘label’ and ‘hint’.</td>
</tr>
</tbody>
</table>

**ServiceNow Perl API examples**

This page provides examples for use of the ServiceNow Perl API.

© 2020 ServiceNow, Inc. All rights reserved.

ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Creating an Incident

#!/usr/bin/perl -w
use ServiceNow;
use ServiceNow::Configuration;
my $CONFIG = ServiceNow::Configuration->new();
$CONFIG->setSoapEndPoint("https://demoi1.service-now.com/");
$CONFIG->setUserName("admin");
$CONFIG->setUserPassword("admin");
my $SN = ServiceNow->new($CONFIG);
my $number = $SN->createIncident({"short_description" => "this incident was created from the Perl API", "category" => "hardware"});
print $number ."\n";

Querying an Incident

#!/usr/bin/perl -w
use ServiceNow;
use ServiceNow::Configuration;
my $CONFIG = ServiceNow::Configuration->new();
$CONFIG->setSoapEndPoint("https://demoi1.service-now.com/");
$CONFIG->setUserName("admin");
$CONFIG->setUserPassword("admin");
my $SN = ServiceNow->new($CONFIG);
my @incidents = $SN->queryIncident({'number' => 'INC00002'});
my $count = scalar(@incidents);
print "number of incidents= " . $count . "\n";
foreach my $incident (@incidents) {
    print "Incident number: $incident->{'number'}\n";
    print "Assignent Group: $incident->{'assignment_group'}\n";
    print "Opened by: $incident->{'opened_by'}\n";
    print "Opened by DV: $incident->{'dv_opened_by'}\n";
    print "SD: $incident->{'short_description'}\n";
    print "TW: $incident->{'time_worked'}\n"
    print "\n";
}

Querying Journal fields

#!/usr/bin/perl -w
use ServiceNow;
use ServiceNow::Configuration;
my $CONFIG = ServiceNow::Configuration->new();
```perl
$CONFIG->setSoapEndPoint("https://demoi1.service-now.com/");
$CONFIG->setUserName("admin");
$CONFIG->setUserPassword("admin");

my $SN = ServiceNow->new($CONFIG);

my @journals = $SN->queryJournal('INC00002');

print $journals[0]->{'element'} . " = " . $journals[0]->{'value'} . "\n";
print $journals[1]->{'element'} . " = " . $journals[1]->{'value'} . "\n";
```

### Perl API GlideRecord

ServiceNow Perl API - GlideRecord perl module is an object representation of a GlideRecord object used to access your ServiceNow instance.

#### Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>new(Configuration object, Table name, optional caller object)</td>
</tr>
<tr>
<td></td>
<td>$config = ServiceNow::Configuration-&gt;new();</td>
</tr>
<tr>
<td></td>
<td>$glideRecord = ServiceNow::GlideRecord-&gt;new($config,'incident',$me);</td>
</tr>
</tbody>
</table>

Constructor. Access to the ServiceNow Glide Record object. The caller object is optional unless creating a Class that inherits GlideRecord (See any class in ServiceNow/ITIL for example).

#### Subroutines

<table>
<thead>
<tr>
<th>Subroutines</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert</td>
<td>insert(optional hash argument)</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>$glideRecord-&gt;insert();</td>
</tr>
<tr>
<td></td>
<td>Inserts glide record into Table. Returns sys id.</td>
</tr>
<tr>
<td>setValue</td>
<td>setValue(name, value)</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>$glideRecord-&gt;setValue('caller_id','56');</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
<table>
<thead>
<tr>
<th>Subroutines</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets element within Glide Record with name to specified value. Will not effect the GlideRecord within the Table until inserted or updated.</td>
</tr>
<tr>
<td>addQuery</td>
<td>addQuery(name, value)</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>$glideRecord-&gt;addQuery('number','INC1000014');</td>
</tr>
<tr>
<td></td>
<td>$glideRecord-&gt;query();</td>
</tr>
<tr>
<td></td>
<td>Refines query to include only the Glide Records with field name=value.</td>
</tr>
<tr>
<td>query</td>
<td>query(optional hash arguments)</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>$glideRecord-&gt;query();</td>
</tr>
<tr>
<td></td>
<td>Returns all Glide Records in the Table with specified query. Step through the Records with the next() call.</td>
</tr>
<tr>
<td>next</td>
<td>next()</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>if($glideRecord-&gt;next());</td>
</tr>
<tr>
<td></td>
<td>while($glideRecord-&gt;next());</td>
</tr>
<tr>
<td></td>
<td>Steps through the results of Glide Record query. Returns TRUE if more elements exist.</td>
</tr>
<tr>
<td>update</td>
<td>update(optional hash arguments)</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>$glideRecord-&gt;setValue('name','value');</td>
</tr>
<tr>
<td></td>
<td>$glideRecord-&gt;update();</td>
</tr>
<tr>
<td></td>
<td>Updates Glide Record in table with the Glide Record object. Changes to Glide Record object will not take effect until updated or inserted. Returns sys_id of record on success, undef of failure.</td>
</tr>
<tr>
<td>getValue</td>
<td>getValue(name)</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>$glideRecord-&gt;getValue($name);</td>
</tr>
<tr>
<td></td>
<td>Get value of element name in GlideRecord. Returns string value of element.</td>
</tr>
<tr>
<td>getDisplayValue</td>
<td>getDisplayValue(name)</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
### Subroutines

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>create(optional parameters)</td>
<td>$task-&gt;create();</td>
</tr>
<tr>
<td>close</td>
<td>close(number of record, optional parameters)</td>
<td>$task-&gt;close('INC1000312');</td>
</tr>
</tbody>
</table>

#### Perl API task

ServiceNow Perl API - task module is an object representation of a Task in the ServiceNow platform.

The Perl API task provides subroutines for querying, updating, and creating tasks. Task is the parent class of Incident, Problem, Change, SC_Task and Ticket. These child classes inherit subroutines from this class.

#### Constructor

new(Configuration);

Example:

```perl
$task = ServiceNow::ITIL::Task->new($CONFIG);
```

This example takes a configuration object and manufactures a Task object connected to the ServiceNow instance.

#### Perl API task subroutines

The Perl API task provides subroutines for querying, updating, and creating tasks.

### Description

Example:

```perl
$glideRecord->getDisplayValue($name);
```

Gets display value of element name in GlideRecord. A display value would be the string name, instead of the sys_id in the case of a reference field, or the string value instead of the number value in the case of choice fields.
<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>reopen</td>
<td>reopen(number of record, optional parameters)</td>
<td><img src="Example_code_reopen.png" alt="" /></td>
</tr>
<tr>
<td>reassign</td>
<td>reassign(number, group, user)</td>
<td><img src="Example_code_reassign.png" alt="" /></td>
</tr>
<tr>
<td>queryJournal</td>
<td>queryJournal(configuration file, optional field name)</td>
<td><img src="Example_code_queryJournal.png" alt="" /></td>
</tr>
<tr>
<td>attach</td>
<td>attach(file path)</td>
<td><img src="Example_code_attach.png" alt="" /></td>
</tr>
</tbody>
</table>

### Perl API Incident

An object representation of an incident in the ServiceNow platform. Provides subroutines for querying, updating, and creating incidents.

### Incident module

ServiceNow Perl API - Incident perl module

### Creating an Incident

```perl
#!/usr/bin/perl -w
use ServiceNow;
use ServiceNow::Configuration;
use ServiceNow::ITIL::Incident;
my $CONFIG = ServiceNow::Configuration->new();
```
$CONFIG->setSoapEndPoint("https://demoi1.service-now.com/");
$CONFIG->setUserName("admin");
$CONFIG->setUserPassword("admin");

# setting incident values as a hash map in the insert argument
my $incident = ServiceNow::ITIL::Incident->new($CONFIG);
my $sys_id = $incident->insert({"short_description" => "this incident was created from the Perl API", "category" => "hardware"]);
print $sys_id ."\n";

# setting incident values by making setValue calls to the incident object
$incident = ServiceNow::ITIL::Incident->new($CONFIG);
$incident->setValue("short_description", "this incident was created from the Perl API - 2");
$incident->setValue("category", "hardware");
$sys_id = $incident->insert();
print $sys_id ."\n";

---

**Querying for Incidents**

```
#!/usr/bin/perl -w
use ServiceNow;
use ServiceNow::Configuration;
use ServiceNow::ITIL::Incident;
my $CONFIG = ServiceNow::Configuration->new();
$CONFIG->setSoapEndPoint("https://demoi1.service-now.com/");
$CONFIG->setUserName("admin");
$CONFIG->setUserPassword("admin");

my $incident = ServiceNow::ITIL::Incident->new($CONFIG);
$incident->addQuery("assignment_group", "Service Desk");
$incident->addQuery("category", "Hardware");
$incident->query();
while($incident->next()) {
    print "number=" . $incident->getValue("number") . "\n";
    print "sd=" . $incident->getValue("short_description") . "\n";
    print "opened_by Display Value= " . $incident->getDisplayValue("opened_by") . "\n";
    print "opened_by sys_id= " . $incident->getValue('opened_by');
}
```

---

**Adding an attachment to a newly created Incident**

```
#!/usr/bin/perl -w
use ServiceNow;
use ServiceNow::Configuration;
use ServiceNow::ITIL::Incident;
my $CONFIG = ServiceNow::Configuration->new();
$CONFIG->setSoapEndPoint("https://demoi1.service-now.com/");
$CONFIG->setUserName("admin");
```
$CONFIG->setUserPassword("admin");

my $incident = ServiceNow::ITIL::Incident->new($CONFIG);
$incident->setValue("short_description", "test incident for attachment 2");
$incident->insert();
$incident->attach("/Users/davidloo/Desktop/test_files/number_test.xls");

Constructor

new

new(Configuration);
Example:

$incident = ServiceNow::ITIL::Incident->new($CONFIG);

Takes a configuration object and manufactures an Incident object connected to the ServiceNow instance.

Subroutines inherited from Task.pm

- attach
- close
- create
- queryJournal
- reassign
- reopen

Subroutines inherited from GlideRecord.pm

- addQuery
- getValue
- getDisplayValue
- setValue
- next
- insert
- query
- update

Subroutines

close

close(number, hashmap);
Example:

$incident->close($number)

Close an incident and update values described in the hash map passed in.

reopen

reopen(number, hashmap);

Example:

$incident->reopen($number);

Re-open a closed incident and update values described in the hash map passed in.

createProblem

createProblem();

Create a problem ticket from an incident and associate it. Returns the sys_id of the newly created problem ticket.

createChange

createChange();

Create a change request from an incident and associate it. Returns the sys_id of the newly created change request.

Perl API change

An object representation of a change in the ServiceNow platform. Provides subroutines for querying, updating, and creating change requests.

Constructor

new

new(Configuration);

Example:

$change = ServiceNow::ITIL::Change->new($CONFIG);

Takes a configuration object and manufactures an Change object connected to the ServiceNow instance.

Subroutines inherited from Task.pm

- attach
- close
- create
- queryJournal
- reassign
Subroutines inherited from GlideRecord.pm

- addQuery
- getValue
- getDisplayValue
- setValue
- next
- insert
- query
- update

Perl API problem
An object representation of a problem in the ServiceNow platform. Provides subroutines for querying, updating, and creating problems.

Constructor - new

new(Configuration);

Example:

```perl
$problem = ServiceNow::ITIL::Problem->new($CONFIG);
```

Takes a configuration object and manufactures a Problem object connected to the ServiceNow instance.

Subroutines inherited from Task.pm

- attach
- close
- create
- queryJournal
- reassign
- reopen

Subroutines inherited from GlideRecord.pm

- addQuery
- getValue
- getDisplayValue
- setValue
- next
- insert
- query
- update
Perl API request
An object representation of a request in the ServiceNow platform. Provides subroutines for querying, updating, and creating service catalog requests.

Description
An object representation of an Request in the ServiceNow platform. Provides subroutines for querying, updating, and creating service catalog requests.

Constructor

class
new
new(Configuration);

Example:

```perl
$request = ServiceNow::ITIL::Request->new($CONFIG);
```

Takes a configuration object and manufactures an Request object connected to the ServiceNow instance.

Subroutines inherited from Task.pm

- attach
- close
- create
- queryJournal
- reassign
- reopen

Subroutines inherited from GlideRecord.pm

- addQuery
- getValue
- getDisplayValue
- setValue
- next
- insert
- query
- update

Subroutines

createRequest

createRequest(user);
Create a service request for the specified user

Perl API requested item

<table>
<thead>
<tr>
<th>ServiceNow Perl API Module</th>
<th>Description</th>
<th>Constructor</th>
<th>Subroutines inherited from Task.pm</th>
<th>Subroutines inherited from GlideRecord.pm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequestedItem</td>
<td>An object representation of a Requested Item in the ServiceNow platform. Provides subroutines for querying, updating, and creating Service Catalog requested item. System Requirements The ServiceNow Perl API requires Perl 5.8 with the following modules installed: • SOAP::Lite (prerequisites <a href="http://soaplite.com/prereqs.html">http://soaplite.com/prereqs.html</a>) 0.71 or later • Crypt::SSLeay • IO::Socket::SSL</td>
<td>new(Configuration); Example: $req_item = ServiceNow::ITIL::RequestedItem-&gt;new($CONFIG); Takes a configuration object and manufactures a Requested Item object connected to the ServiceNow instance</td>
<td>• attach • close • create • queryJournal • reassign • reopen</td>
<td>• addQuery • getDisplayValue • getDisplayValue • attach • close • create • queryJournal • reassign • reopen</td>
</tr>
</tbody>
</table>

Perl API dictionary
An object representation of a dictionary record in the ServiceNow platform. Provides subroutines for querying, updating, and creating sys_dictionary.

Dictionary module
ServiceNow Perl API - dictionary module

new

new(Configuration);

Example:

$dic = ServiceNow::ITIL::Dictionary->new($CONFIG);

Takes a configuration object and manufactures a Dictionary object connected to the ServiceNow instance
**queryFields**

queryFields(table name, optional boolean choice)

Example:

```
$sys_dictionary->queryFields('incident');
```

Returns an array of hashes containing each field for specified table. If choice is specified then get choice values for each of the choice fields and put them into the hash element 'choices'.

**Perl API configuration**

<table>
<thead>
<tr>
<th>ServiceNow Perl API Module</th>
<th>Description</th>
<th>Constructor</th>
</tr>
</thead>
</table>
| Ticket Perl               | An object representation of a Configuration object used to access your ServiceNow instance. | `new();` Example: 
  `$conf = ServiceNow::Configuration->new();` |
|                           | System Requirements | Create a new Configuration object and start customizing it to be used for other objects. |
|                           | The ServiceNow Perl API requires Perl 5.8 with the following modules installed: | |
|                           | · SOAP::Lite (prerequisites [http://soaplite.com/prereqs.html](http://soaplite.com/prereqs.html)) 0.71 or later | |
|                           | · Crypt::SSLeay | |
|                           | · IO::Socket::SSL | |
### Subroutines

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getSoapEndPoint</td>
<td>Gets the complete SOAP endpoint used to access your ServiceNow instance, given the table name.</td>
</tr>
<tr>
<td>setSoapEndPoint</td>
<td>Sets the complete SOAP endpoint used to access your ServiceNow instance.</td>
</tr>
<tr>
<td>getUserName</td>
<td>Get the user name used to authenticate a connection to the SOAP endpoint.</td>
</tr>
<tr>
<td>setUserName</td>
<td>Set the user name used to authenticate a connection to the SOAP endpoint.</td>
</tr>
<tr>
<td>getUserPassword</td>
<td>Get the user password used to authenticate a connection to the SOAP endpoint.</td>
</tr>
<tr>
<td>setUserPassword</td>
<td>Set the user password used to authenticate a connection to the SOAP endpoint.</td>
</tr>
<tr>
<td>getConnection</td>
<td>Get the Connection object used to access the ServiceNow SOAP endpoint.</td>
</tr>
</tbody>
</table>

**Perl API attachment**

The ServiceNow Perl API - Attachment perl module is an object representation of an Attachment in the ServiceNow platform. Provides subroutines for creating an attachment and attaching to an existing record.

**Constructor - new**

new(Configuration);

Example:

```perl
$task = ServiceNow::Attachment->new($CONFIG);
```

Takes a configuration object and manufactures an Task object connected to the ServiceNow instance.

**Subroutines - create**

create(path, table_name, sys_id)
Example:

```
$attachment->create("/Users/davidloo/Desktop/test_files/number_test.xls", "incident", "9d385017c611228701d22104cc95c371");
```

Creates an attachment from a file on the local disk, to an existing record defined by table_name and sys_id Returns the sys_id of the ecc_queue record, undef if failed

**Perl API service catalog task**

<table>
<thead>
<tr>
<th>ServiceNow Perl API Module</th>
<th>Description</th>
<th>Constructor</th>
<th>Subroutines inherited from Task.pm</th>
<th>Subroutines inherited from GlideRecord.pm</th>
</tr>
</thead>
</table>
| SC_Task                    | An object representation of an Service Request Task in the Now Platform. Provides subroutines for querying, updating, and creating sc_task. System Requirements The ServiceNow Perl API requires Perl 5.8 with the following modules installed:  
  - SOAP::Lite (prerequisites http://soaplite.com/prereqs.html) 0.71 or later  
  - Crypt::SSLeay  
  - IO::Socket::SSL | new(Configuration); Example:  
  ```perl
  $sc_task = ServiceNow::ITIL::RequestedItem->new($CONFIG);
  ```  
  Takes a configuration object and manufactures a Service Catalog Task object connected to the ServiceNow instance |  
  - attach  
  - close  
  - create  
  - queryJournal  
  - reassign  
  - reopen |  
  - addQuery  
  - getValue  
  - getDisplayValue  
  - setValue  
  - next  
  - insert  
  - query  
  - update |

**Perl API ticket**

<table>
<thead>
<tr>
<th>ServiceNow Perl API Module</th>
<th>Description</th>
<th>Constructor</th>
<th>Subroutines inherited from Task.pm</th>
<th>Subroutines inherited from GlideRecord.pm</th>
</tr>
</thead>
</table>
| Ticket                     | An object representation of a Ticket in the ServiceNow platform. Provides subroutines for querying, updating, and creating tickets. | new(Configuration); Example:  
  ```perl
  $ticket = ServiceNow::ITIL::Ticket->new($CONFIG);
  ```  
  Takes a configuration object and manufactures a Ticket object and manufactures |  
  - attach  
  - close  
  - create  
  - queryJournal  
  - reassign  
  - reopen |  
  - addQuery  
  - getValue  
  - getDisplayValue  
  - setValue  
  - next  
  - insert  
  - query  
  - update |
<table>
<thead>
<tr>
<th>ServiceNow Perl API Module</th>
<th>Description</th>
<th>Constructor</th>
<th>Subroutines inherited from</th>
<th>Subroutines inherited from</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>System Requirements</td>
<td>Constructor</td>
<td>glideRecord.pm</td>
<td>Task.pm</td>
</tr>
<tr>
<td></td>
<td>The ServiceNow Perl API requires Perl 5.8 with the following modules installed:</td>
<td>a Ticket object connected to the ServiceNow instance</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· SOAP::Lite (prerequisites <a href="http://soaplite.com/prereqs.html">http://soaplite.com/prereqs.html</a>) 0.71 or later</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· Crypt::SSLeay</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· IO::Socket::SSL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Perl API approval**

<table>
<thead>
<tr>
<th>ServiceNow Perl API Module</th>
<th>Description</th>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approval</td>
<td>An object representation of an Approval record in the ServiceNow instance. Provides subroutines for querying, updating, and creating approvals.</td>
<td>new(Configuration);</td>
</tr>
<tr>
<td></td>
<td>System Requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The ServiceNow Perl API requires Perl 5.8 with the following modules installed:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>· SOAP::Lite (prerequisites <a href="http://soaplite.com/prereqs.html">http://soaplite.com/prereqs.html</a>) 0.71 or later</td>
<td></td>
</tr>
<tr>
<td></td>
<td>· Crypt::SSLeay</td>
<td></td>
</tr>
<tr>
<td></td>
<td>· IO::Socket::SSL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$approvals =</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ServiceNow::ITIL::Approval- &gt;new($CONFIG);</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Takes a configuration object and manufactures an Approval object connected to the ServiceNow instance.</td>
<td></td>
</tr>
</tbody>
</table>
### Perl API examples

This reference provides examples of the Perl API.

#### Using the ServiceNow.pm module

**Creating an incident:**

```perl
#!/usr/bin/perl -w
use ServiceNow;
use ServiceNow::Configuration;

my $CONFIG = ServiceNow::Configuration->new();

$CONFIG->setSoapEndPoint("https://<instance name>.service-now.com/");
$CONFIG->setUserName("admin");
$CONFIG->setUserPassword("admin");

my $SN = ServiceNow->new($CONFIG);
my $number = $SN->createIncident({"short_description" => "this incident was created from the Perl API", "category" => "hardware"]);

print $number ."\n";
```

**Querying an Incident:**

```perl
#!/usr/bin/perl -w
use ServiceNow;
use ServiceNow::Configuration;

my $CONFIG = ServiceNow::Configuration->new();

$CONFIG->setSoapEndPoint("https://<instance name>.service-now.com/");
```
$CONFIG->setUserName("admin");
$CONFIG->setUserPassword("admin");

my $SN = ServiceNow->new($CONFIG);

my @incidents = $SN->queryIncident({'number' => 'INC00002'});
my $count = scalar(@incidents);
print "number of incidents=\$count\n";
foreach my $incident (@incidents) {
  print "Incident number: \$incident->'number'\n";
  print "Assignent Group: \$incident->'assignment_group'\n";
  print "Opened by: \$incident->'opened_by'\n";
  print "Opened by DV: \$incident->'dv_opened_by'\n";
  print "SD: \$incident->'short_description'\n";
  print "TW: \$incident->'time_worked'\n";
  print "\n";
}

Querying Journal fields:

#!/usr/bin/perl -w
close ServiceNow;
close ServiceNow::Configuration;
my $CONFIG = ServiceNow::Configuration->new();

$CONFIG->setSoapEndPoint("https://<instance name>.service-now.com/");
$CONFIG->setUserName("admin");
$CONFIG->setUserPassword("admin");

my $SN = ServiceNow->new($CONFIG);
my @journals = $SN->queryJournal('INC00002');
print $journals[0]->'element' . " = " . $journals[0]->'value' . "\n";
print $journals[1]->'element' . " = " . $journals[1]->'value' . "\n";

Using the ITIL Objects

Creating an Incident:

#!/usr/bin/perl -w
use ServiceNow;
use ServiceNow::Configuration;
use ServiceNow::ITIL::Incident;
my $CONFIG = ServiceNow::Configuration->new();

$CONFIG->setSoapEndPoint("https://<instance name>.service-now.com/");
$CONFIG->setUserName("admin");
$CONFIG->setUserPassword("admin");

my $incident = ServiceNow::ITIL::Incident->new($CONFIG);
my $sys_id = $incident->insert({"short_description" => "this incident was created from the Perl API", "category" => "hardware"});
print $sys_id . "\n";

# setting incident values by making setValue calls to the incident object
$incident = ServiceNow::ITIL::Incident->new($CONFIG);
$incident->setValue("short_description", "this incident was created from the Perl API - 2");
$incident->setValue("category", "hardware");
$sys_id = $incident->insert();
print $sys_id ."\n";

### Querying for Incidents:

```perl
#!/usr/bin/perl -w
use ServiceNow;
use ServiceNow::Configuration;
use ServiceNow::ITIL::Incident;

my $CONFIG = ServiceNow::Configuration->new();
$CONFIG->setSoapEndPoint("https://<instance name>.service-now.com/");
$CONFIG->setUserName("admin");
$CONFIG->setUserPassword("admin");

my $incident = ServiceNow::ITIL::Incident->new($CONFIG);
$incident->addQuery("assignment_group", "Service Desk");
$incident->addQuery("category", "Hardware");
$incident->query();
while($incident->next()) {
    print "number= " . $incident->getValue("number") . "\n";
    print "sd= " . $incident->getValue("short_description") . "\n";
    print "opened_by Display Value= " . $incident->getDisplayValue("opened_by") . "\n";
    print "opened_by sys_id= " . $incident->getValue('opened_by');
}
```

### Adding an attachment to a newly created Incident:

```perl
#!/usr/bin/perl -w
use ServiceNow;
use ServiceNow::Configuration;
use ServiceNow::ITIL::Incident;

my $CONFIG = ServiceNow::Configuration->new();
$CONFIG->setSoapEndPoint("https://<instance name>.service-now.com/");
$CONFIG->setUserName("admin");
$CONFIG->setUserPassword("admin");

my $incident = ServiceNow::ITIL::Incident->new($CONFIG);
$incident->setValue("short_description", "test incident for attachment 2");
$incident->insert();
$incident->attach("/Users/davidloo/Desktop/test_files/number_test.xls");
```

## Scripted SOAP web services

Scripted SOAP Web Services allow a ServiceNow administrator to create new web services that are not addressed by the system.

You can define input and output parameters for the web service and use JavaScript to perform operations. Though this feature is very powerful, use [Direct Web Services](https://service-now.com/) or [Web Service Import sets](https://service-now.com/) instead whenever possible since they are simpler to implement and maintain.
Security

When *strict security* is enforced on a system, the HTTP authenticated user must have the *soap_script* role to execute the scripted web service.

Creating a new web service

When the Web Services Provider - Scripted plugin is activated, a new module *Scripted Web Services* is available under the *System Web Services* application.

Scripted Web Services module

Click the module to display a list of example scripted Web Services.
Scripted SOAP Web Services

Example 1: Retrieving a system property

The first step is to define the incoming and return parameters. This is done by adding an entry to the **Input Parameters** and **Output Parameters**. These parameters are used to construct and present a meaningful WSDL, and they do not add to the functionality of processing the actual Web Service itself.

GetProperty Input & Output Parameters

The parameters are referenced in the script of the Web Service. Any of the input parameters are retrieved using the following syntax:

```javascript
var a = request.property;
```
The output parameters are set by using the following syntax:

```java
response.property="ABC";
```

The following example demonstrates how to retrieve a system property and return it as part of the SOAP response. The example shows how to create a custom scripted Web Service to do something specific that the base ServiceNow system direct Web Services cannot.

### GetProperty web service

**Example 2: Ordering a Blackberry**

Direct Web Services in ServiceNow operate on tables and their data, while the following example shows how to initiate a business solution, such as ordering a Blackberry, by invoking a scripted Web Service. The following input and output parameters will support the Blackberry example:
This script shows how to use the above parameters to add a Blackberry to the service catalog shopping cart and order it. The request number is returned in the `request_number` field of the SOAP response.

```javascript
var cart = new Cart();
var item = cart.addItem('e2132865c0a8016500108d9cee411699');
cart.setVariable(item, 'original', request.phone_number);

// set the requested for
var gr = new GlideRecord("sys_user");
gr.addQuery("user_name", request.requested_for);
gr.query();
if(gr.next()) {
    var cartGR = cart.getCart();
    cartGR.requested_for = gr.sys_id;
    cartGR.update();
}

var rc = cart.placeOrder();
response.request_number = rc.number;
```

**Global Variables**

To facilitate custom processing of incoming SOAP requests, the following global variables are available in the script context:

2. `soapRequestXML`: a string object representing the incoming SOAP envelope XML.
3. `request`: a Javascript object containing mapped values (mapped to input parameter names) of the incoming SOAP envelope.
4. `response`: a Javascript object which allows the script programmer to customize the response values. See `Customize Response`
Customize response

Follow this example in order to customize and have control over the XML payload of the SOAP response.

1. Create a customized XML document using the `XMLDocument` script include object.

   Note: When creating a scripted web service in a scoped application you must use the API.

2. Set its document element to the variable `response.soapResponseElement` in a scripted web service.

For example, the following scripted web service script:

```javascript
var xmldoc = new XMLDocument2();
xmldoc.parseXML("<myResponse></myResponse>");
xmldoc.createElementWithTextValue("element_one", "test");
xmldoc.createElementWithTextValue("element_two", "new2 value");

var el = xmldoc.createElement("element_three");
xmldoc.setCurrentElement(el);
xmldoc.createElementWithTextValue("newChild", "test child element");
response.soapResponseElement = xmldoc.getDocumentElement();
```

Is used to accept the following request:

```xml
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
  xmlns:tes="http://www.service-now.com/TestCustomResponse">
  <soapenv:Header/>
  <soapenv:Body>
    <tes:execute/>
  </soapenv:Body>
</soapenv:Envelope>
```

Which will respond with the following SOAP response:

```xml
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
  xmlns:tes="http://www.service-now.com/TestCustomResponse">
  <soapenv:Header/>
  <soapenv:Body>
    <myResponse>
      <element_one>test</element_one>
      <element_two>new2 value</element_two>
      <element_three>
        <newChild>test child element</newChild>
      </element_three>
    </myResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

WSDL support will need to be created externally. The SOAP endpoint will need to be referred back to the scripted web service in question.

**Static WSDL**

A static WSDL allows you to define a custom WSDL for web service requests.
Certain web service clients require SOAP access to ServiceNow through a specific Web Service Description Language (WSDL) format. This required format may differ from the standard ServiceNow WSDL format. In these cases you can create a static WSDL that matches the required format.

Create a scripted web service
To use a static WSDL, create a scripted web service.

Role required: web_service_admin or admin

1. Navigate to **System Web Service** > **Scripted Web Services**.
2. Click **New**.
3. Enter a **Name** for the scripted web service such as FakeStockValue.
4. Enter a **Script** for the web service to run.
5. Click **Submit**.

**Scripted web service example**
This example demonstrates the processing script for the FakeStockValue web service.

```javascript
var vProcessor = new FakeStockValue(soapRequestXML);

var responseElement = vProcessor.process();
if (responseElement != null) {
    response.soapResponseElement = responseElement;
} else {
    response.soapResponseElement = vProcessor.generateSoapFault("unknown error");
}
```

Create a static WSDL
Create a static WSDL with the required format to override the standard WSDL for your scripted web service.

Role required: web_service_admin or admin

1. Navigate to **System Web Services** > **Static WSDL**.
2. Create a static WSDL record using the same name as the scripted web service, such as FakeStockValue.
3. Enter the custom WSDL into the **WSDL** field.
4. Click **Submit**.

**Static WSDL example**
This example demonstrates the FakeStockValue WSDL.

```xml
<?xml version= "1.0" ?>

<types><schema targetNamespace = "http://example.com/stockquote.xsd" xmlns = "http://www.w3.org/2000/10/XMLSchema" ><element name = "TradePriceRequest" ><complexType><all><element name = "tickerSymbol" type = "string" /></all></complexType></element><element name = "TradePrice" ><complexType><all><element name = "price" type = "float" /></all></complexType></element></schema></types>

<message name = "GetLastTradePriceInput" ><part name = "body" element = "xsd1:TradePriceRequest" /></message>

<message name = "GetLastTradePriceOutput" ><part name = "body" element = "xsd1:TradePrice" /></message>
```
Create a static WSDL script include
Create a script include to define the majority of the code used to process static WSDL requests.

Role required: script_include_admin or admin

By implementing the majority of the custom functionality in a script include, you can reuse the script include in multiple areas.

1. Navigate to System UI > Script Includes.
2. Click New.
3. Enter a Name for the script include that matches the name of the static WSDL, such as FakeStockValue.
4. Enter the script include code in the Script field.
5. Click Submit.

Static WSDL script include example
This example demonstrates the FakeStockValue script include that implements much of the static WSDL behavior.

```javascript
var FakeStockValue = Class.create();

FakeStockValue.prototype = {
    initialize : function(requestXML) {
        //Use some backend XML utilities...you could use string tools if you wish
        this.xmlutil = Packages.com.glide.util.XMLUtil;
        //converting the string to an XML Document
        this.fSoapDoc = new XMLDocument(requestXML);
    },

    process : function() {
        var soapBody = this.fSoapDoc.getNode("/Envelope/Body");
        //Our WSDL was formatted to have the only first child element be the function
        var funcNode = this.xmlutil.getFirstChildElement(soapBody);
        var nodeName = this.xmlutil.getNodeNameNS(funcNode);

        //If the function for this SOAP request is TradePriceRequest, perform the necessary actions
        if (nodeName == "TradePriceRequest") {
            return this.fakeOutTradePriceRequest(funcNode);
        }

        //Couldn't find any supported functions in this SOAP request
```
return this.generateSoapFault("un-supported API call: " + nodeName);
},

fakeOutTradePriceRequest : function (funcNode) {
    //Create the beginnings of our XML response
    var r = new XMLDocument("<GetLastTradePriceOutput xmlns='https://www.service-now.com/vws/FakeStockValue'/>");

    //Do the necessary actions here...we're going to get the USER ID of the user
    //used to make this SOAP call. Then we will return the
    //stock symbol they were asking about
    var usersysid = gs.getUserID();
    var gr = new GlideRecord("sys_user");
    gr.get(usersysid);
    var username = gr.user_name;
    var quoteSymbol = this.xmlutil.getText(funcNode);
    //Create a "message" element to store our response message
    r.createElement("message", username + ", You were looking for a quote on "+quoteSymbol);
    return r.getDocumentElement();
},

generateSoapFault : function (str) {
    var f = "<SOAP-ENV:Fault>" +
        "<faultcode xsi:type='xsd:string'>SOAP-ENV:FakeStockValue</faultcode>" +
        "<faultstring xsi:type='xsd:string'>" + str +
        "</faultstring>" +
        "</SOAP-ENV:Fault>"
    var s = new XMLDocument(f);
    return s.getDocumentElement();
}

initialize function

The initialize function takes the XML request string and converts it to an XML Document object that you can navigate and manipulate using libraries. Alternatively, you can leave the XML request as a string and navigate it using regular expressions.

process function

The process function is called by the scripted web service. This function grabs the first child element in the XML after the body element. The WSDL uses this child element to determine which function to use. In this WSDL there is only one possible function but most WSDLs provide many functions. If more functions were available, there would be more 'if' statements that tested the first child element for the various function names.

fakeOutTradePriceRequest function

The fakeOutTradePriceRequest function is the implementation of the only available function in the WSDL. This function looks up the user that the SOAP request authenticated as and retrieves the username then returns it to the SOAP client. The fakeOutTradePriceRequest function could
be expanded to perform useful activities, such as looking up a stock symbol and returning the last traded price.

**generateSoapFault function**

The `generateSoapFault` function returns a SOAP error that can be called if there are problems.

*Use the static WSDL*

Load the static WSDL into a SOAP client to make requests to the SOAP web service.

The web service client provides

- The FakeStockValue project.
- The StockQuoteBinding web service.
- The `GetLastTradePrice` SOAP function. This function generates request records when run.

```
Loaded WSDL

You can change the default request XML in the static WSDL to include a stock symbol.

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:stoc="http://example.com/stockquote.xsd">
  <soapenv:Header/>
  <soapenv:Body>
    <stoc:TradePriceRequest>IBM</stoc:TradePriceRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Submitting a SOAP request to this web service endpoint returns the following to the requesting SOAP client.

```
  <SOAP-ENV:Body>
    <GetLastTradePriceOutput xmlns="https://www.service-now.com/vws/FakeStockValue">
      <message>admin2, You were looking for a quote on IBM</message>
    </GetLastTradePriceOutput>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Enforcing WSDL compliance**

You can force the response to list output values in the same order as defined in the WSDL.

When you create a scripted SOAP web service, the generated WSDL is based on the Input Parameters and Output Parameters related lists. The actual SOAP response sent by the scripted service is determined by the Script. This behavior can cause the script to return output values in a different order than defined in the WSDL.
To enforce the order of output parameters as defined in the related list, select the **WSDL Compliance** check box. When this check box is selected, the web service reorders the parameters returned by the script to match the order in the WSDL.

**Note:** If additional response parameters are returned by the script, but are not defined in the **Response Parameters** related list, those parameters are excluded from the response when **WSDL Compliance** is selected.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Param 1</td>
<td>200</td>
</tr>
<tr>
<td>Param 2</td>
<td>300</td>
</tr>
<tr>
<td>Param 3</td>
<td>100</td>
</tr>
</tbody>
</table>

The following is the script that sets values for the defined output parameters. Note that in this example script the parameters are set in a different order than defined in the **Output Parameters** related list. Also note the additional parameter **param4** that is not defined in the related list.

```java
Response.param1 = 1;
Response.param4 = 4;
Response.param3 = 3;
```

When the **WSDL Compliance** check box is false, the SOAP response generated by the script is the following:

```xml
<response>
  <param1>1</param1>
  <param4>4</param4>
  <param3>3</param3>
</response>
```

When the **WSDL Compliance** check box is true, the SOAP response generated by the script is the following:

```xml
<response>
  <param3>3</param3>
  <param1>1</param1>
</response>
```

**Inbound web service examples**

Inbound web service examples demonstrate how to access ServiceNow web services.

**Java Apache Axis2 web services client examples**

Examples demonstrating an integration with Axis2 Version 1.4.
Requirements

- An "elementFormDefault" value of qualified means that an unqualified element is in the default namespace defined on an ancestor. If it is "unqualified" then an unqualified element is in the empty namespace (xmlns=""). The default is "unqualified".
- To Resolve the Axis Client deserialization failure you should go to System Properties > Web Services and uncheck the property that sets the elementFormDefault attribute of the embedded XML schema to the value of unqualified. Save the property setting and regenerate your Axis2 client code if your client code was generated before changing this property.

Element Form Default Property

Java Apache Axis2 web services client examples insert

An example class to insert an incident record.

```java
public class Insert {
    public static void main ( String args [ ] ) { try {
        HttpTransportProperties. Authenticator basicAuthentication = new
        HttpTransportProperties. Authenticator ( ) ;
        basicAuthentication. setUsername ( "admin" ) ;
        basicAuthentication. setPassword ( "admin" ) ;

        ServiceNowStub proxy = new ServiceNowStub ( ) ;
        axis2. transport. http. HTTPConstants. CHUNKED, Boolean. FALSE ) ;
        proxy._getServiceClient ( ). getOptions ( ). setProperty (org.
        apache. axis2. transport. http. HTTPConstants. AUTHENTICATE,
        basicAuthentication ) ;

        ServiceNowStub. Insert inc = new ServiceNowStub. Insert ( ) ;
        ServiceNowStub. InsertResponse resp = new ServiceNowStub. InsertResponse
        ( ) ;

        inc. setAssigned_to ( "Christen Mitchell" ) ;
        inc. setCategory ( "hardware" ) ;
        inc. setPriority ( BigInteger. ONE ) ;
        inc. setDescription ( "The WI_FI in the reception area is down" ) ;
        inc. setCaller_id ( "Joe Employee" ) ;

        resp = proxy. insert ( inc ) ;
    }
}
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Java Apache Axis2 web services client examples update

An example of an Axis Client program that calls the getKeys function to query all incidents where the category is Hardware.

getKeys

A list of `sys_id` is returned as a result:

```java
package com.service_now.www;

public class DemoClient {
    public static void main(String args[]) { try {
        ServiceNowStub proxy = new ServiceNowStub();
        ServiceNowStub.GetKeys getInc = new ServiceNowStub.GetKeys();
        ServiceNowStub.GetKeysResponse resp = new ServiceNowStub.GetKeysResponse();
        getInc.setActive(true);
        getInc.setCategory("hardware");
        proxy._getServiceClient().getOptions().setProperty(org.apache.axis2.transport.http.HTTPConstants.CHUNKED, Boolean.FALSE);
        resp = proxy.getKeys(getInc);
        String[] keys = resp.getSys_id();
        System.out.println("Key: " + keys[0]);
    } catch (Exception e) {
        System.out.println(e.toString());
    }
}
```

getRecords

```java
package com.service_now.www;

import com.service_now.www.ServiceNowStub.GetRecordsResult_type0;

public class GetRecords {
    /**
     * @param args
     */
    public static void main(String[] args) { try {
        ServiceNowStub proxy = new ServiceNowStub();
        ServiceNowStub.GetRecords incidents = new ServiceNowStub.GetRecords();
        ServiceNowStub.GetRecordsResponse result = new ServiceNowStub.GetRecordsResponse();
        incidents.setActive(true);
        incidents.setCategory("hardware");
        incidents.setSys_created_on("> 2009-06-08 10:30:00");
        } catch (Exception e) {
        System.out.println(e.toString());
    }
```
Java Apache Axis2 web services client examples advanced

Examples showing how to construct and use an Axis2 client to consume a ServiceNow Web Service.

Axis is essentially a SOAP engine -- a framework for constructing SOAP processors such as clients, servers, or gateways. The current version of Axis is written in Java. This content is intended for system admins with a light development background in Java. To begin you would need Java JDK version 1.4.2 or higher and Axis2 version 1.0 or higher.

Create a Java Project

This example uses Eclipse SDK Version: 3.4.2 for managing the source code and executing the web request. Eclipse is not required.

- Open Eclipse and from the menu select File > New > Project > Java Project.
- Give the project a name.
- Verify that the correct JRE is specified.
  - If using wsdl2java run "java -version" on the command line and this will be the version to specify for the project specific JRE.
  - If using the Axis2 Codegen plugin use default JRE.
Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: TestWebService

Contents

- Create new project in workspace
- Create project from existing source

Directory: /glide/workspace/TestWebService

JRE

- Use default JRE (Currently 'JVM 1.4.2 (MacOS X Default)')
- Use a project specific JRE: JVM 1.6.0
- Use an execution environment JRE: J2SE-1.4

Project layout

- Use project folder as root for sources and class files
- Create separate folders for sources and class files

Working sets

- Add project to working sets

Working sets: Select...
Generate your Axis2 client code

- From a command line in the bin directory of the axis folder:

```
./wsdl2java.sh -uri https://<instance name>.service-now.com/incident.do?WSDL -o /glide/workspace/TestWebService/
```

- In the above example:
  - The "-uri" is either the path where you have saved a copy of the wsdl to either ".wsdl" or "xml", or the URL the WSDL resides at.
  - The "-o" is the path where you want the files to be written out to. If not specified, the files will be written out to the current bin location.

- In Eclipse refresh the project and the generated Stub and CallbackHandler should now be displayed

```
Axis Stub
```

Basic Authentication

```
HttpTransportProperties.Authenticator basicAuthentication = new HttpTransportProperties.Authenticator ();
basicAuthentication. setUsername ( "admin" ) ;
basicAuthentication. setPassword ( "admin" ) ;
...
ServiceNowStub proxy = new ServiceNowStub () ;
...
proxy._getServiceClient (). getOptions (). setProperty (org.apache.axis2.transport.http.HTTPConstants.AUTHENTICATE,
basicAuthentication ) ;
```

Compatibility with Axis2 Versions 1.1 and higher

Chunking support is only available in HTTP Version 1.1. By default chunking is enabled in Axis2.xml for versions 1.1 and higher. ServiceNow does not support Chunking, so you will need to disable chunking at deployment time or at runtime.
• Deployment time: One can disable HTTP chunking by removing or commenting out the following element from Axis2.xml

```
<parameter name="Transfer-Encoding">chunked</parameter>
```

• Runtime: User can disable the chunking using following property set in Client or Stub, versions 1.1.1 and higher only

```
options.setProperty (org.apache.axis2.transport.http.HTTPConstants.CHUNKED, Boolean.FALSE);
```

Creating Unique Packages

You can use the Axis2 parameter namespace2package (ns2p) to create unique package names. The parameter uses this format:

```
<Axis path>\bin\wsdl2java.bat -u -p cr2 -ns2p <namespace>=<package name> -uri <wsdl to convert>
```

For example:

```
<Axis path>\bin\wsdl2java.bat -u -p cr2 -ns2p http://www.service-now.com/change_request=my.change_request -uri change_request
```

Microsoft .NET web services client examples

Examples demonstrating an integration with Microsoft .NET Web Services Client.

Requirements

.NET 2.0 Versions and Higher:

• An "elementFormDefault" value of qualified means that an unqualified element is in the default namespace defined on an ancestor. If it is "unqualified" then an unqualified element is in the empty namespace (xmlns=""). The default is "unqualified".

• To Resolve the .NET Client deserialization failure you should go to System Properties > Web Services and uncheck the property that sets the elementFormDefault attribute of the embedded XML schema to the value of unqualified. Save the property setting and recreate your WSDL Reference.cs class. See Also "Compatibility with Clients generated from WSDL" below.
Element Form Default Property

Example insert using Visual Basic .NET

A sample Visual Basic .NET program that inserts a core_company record.

```vbnet
Public Class Class1
    Shared Sub Main()
        Dim proxyCompany As New core_company.localhost.ServiceNow()
        Dim companyInsert As New core_company.localhost.insert()
        Dim companyInsertResponse As core_company.localhost.insertResponse
        With companyInsert
            .name = "Test Company SKF 2"
            .contact = "SKF"
            .customer = True
            .customerSpecified = True
        End With
        companyInsertResponse = proxyCompany.insert(companyInsert)
        Console.WriteLine(companyInsertResponse.sys_id)
    End Sub
End Class
```

Sample Visual Basic .NET project

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Perl web services client examples

Examples demonstrating an integration with a Perl web services client.

**Note:** The following examples require the usage of the *Perl language* and the *SOAP::Lite package*.

System Requirements

- Perl 5.8
  - SOAP::Lite (prerequisites [http://soaplite.com/prereqs.html](http://soaplite.com/prereqs.html))
  - Crypt::SSLeay
  - IO::Socket::SSL

insert

The following example will insert a record into the Incident table.

```perl
#!/usr/bin/perl -w

# declare usage of SOAP::Liteselective SOAP::Lite;

# specifying this subroutine, causes basic auth to use# its credentials when# challenged sub SOAP::Transport::HTTP::Client::get_basic_credentials{
# login as the itil userreturn 'itil' =>'itil';

# declare the SOAP endpoint heremy$soap = SOAP::Lite->proxy('https://
# myinstance.service-now.com/incident.do?SOAP');

# calling the insert functionmy$method = SOAP::Data->name('insert')-
>attr({xmlns =>'http://www.service-now.com/'});

# create a new incident with the following short_description and
categorymy@params = ( SOAP::Data->name(short_description =>'This is an
>category =>$params, SOAP::Data->name(category
>attribute

# invoke the SOAP callmy$result = $soap->call($method=>@params);

# print any SOAP faults that get returnedprint_fault($result);

# convenient subroutine for printing all resultssub print_result
(my($result)=@_;

    if($result->body)&&$result->body->('insertResponse')){
my%keyHash = %($result-
>body->('insertResponse'));foreach my$k (keys%keyHash)
print"name="$k value=

    # convenient subroutine for printing all SOAP faultssub print_fault
(my($result)=@_;
```
insert (With XML payload)

The following is an example of inserting a record into the ecc_queue table where the payload field is an XML document. This is done using the Perl language and the SOAP::Lite package, the XML document creation uses the XML::Writer package:

```perl
#!/usr/bin/perl -w
use SOAP::Lite ( +trace => all, maptype => {} );
use SOAP::Lite;
use XML::Writer;
use XML::Writer::String;

## Get parameters passed by OVO notification
$OVMSG{id}=$ARGV[0];
$OVMSG{node_name}=$ARGV[1];
$OVMSG{node_type}=$ARGV[2];
$OVMSG{date_created}=$ARGV[3];
$OVMSG{time_created}=$ARGV[4];
$OVMSG{date_received}=$ARGV[5];
$OVMSG{time_received}=$ARGV[6];
$OVMSG{application}=$ARGV[7];
$OVMSG{msg_group}=$ARGV[8];
$OVMSG{object}=$ARGV[9];
$OVMSG{severity}=$ARGV[10];
$OVMSG{operator_list}=$ARGV[11];
$OVMSG{msg_text}=$ARGV[12];
$OVMSG{instruction}=$ARGV[13];

sub SOAP::Transport::HTTP::Client::get_basic_credentials{return 'itil' => 'itil';}

my $soap = SOAP::Lite->proxy('http://<instance name>.service-now.com/ecc_queue.do?SOAP');

my $method = SOAP::Data->name('insert')->attr({xmlns => 'http://www.service-now.com/'});

# get all incidents with category Network
my @params = ( SOAP::Data->name(agent =>'OVO_Notification') );
push(@params, SOAP::Data->name(queue => 'input') );
push(@params, SOAP::Data->name(name =>'HP Openview OVO Notification'));
push(@params, SOAP::Data->name(source =>$OVMSG{id}) );

my $s = XML::Writer::String->new();
my $writer=new XML::Writer(OUTPUT =>$s);

#$writer->xmlDecl();
$writer->startTag('notification');

write_element('id');
write_element('node_name');
write_element('node_type');
write_element('date_created');
write_element('time_created');
write_element('date_received');
write_element('time_received');
write_element('application');
write_element('msg_group');
write_element('object');
write_element('severity');
write_element('operator_list');
write_element('msg_text');
write_element('instruction');

$writer->endTag('notification');

$writer->end;

sub write_element {my $label=shift;my $value=$OVMSG{$label};
$writer->startTag($label);
if($value){$writer->characters($value);$writer->endTag($label);}}
```
push(@params, SOAP::Data->name(payload => $s->value()));

print$soap->call($method=>@params)->result;

=== Response to the "insert" ===

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Body>
        <insertResponse xmlns="http://www.service-now.com/ecc_queue">
            <sys_id>1a5ad50e0a0a021101bef2e07705f87a</sys_id>
            <name>HP Openview OVO Notification</name>
        </insertResponse>
    </soap:Body>
</soap:Envelope>

update

#!/usr/bin/perl -w
#use SOAP::Lite ( +trace => all, maptype => {} ); use SOAP::Lite;

sub SOAP::Transport::HTTP::Client::get_basic_credentials { return 'itil' => 'itil'; }

my $soap = SOAP::Lite->proxy('http://localhost:8080/glide/incident.do?SOAP');

my $method = SOAP::Data->name('update') -> attr({ xmlns => 'http://www.service-now.com/' });

# update incident by sys_id
my @params = ( SOAP::Data->name(sys_id => 'e8caecedbc0a80164017df472f39eade1') );
push(@params, SOAP::Data->name(short_description => 'this is a new description'));

my $result = $soap->call($method => @params);

print_fault($result);
print_result($result);

sub print_result { my ($result) = @_;
    if ($result->body && $result->body->{'updateResponse'}) {
        my %keyHash = %{$result->body->{'updateResponse'}};
        foreach my $k (keys %keyHash) {
            print "name=$k value=$keyHash{$k}\n";
        }
    }
}

sub print_fault { my ($result) = @_;
    if ($result->fault) {
        print "faultcode=$result->fault->{'faultcode'}\n";
        print "faultstring=$result->fault->{'faultstring'}\n";
        print "detail=$result->fault->{'detail'}\n";
    }
}

goingKeys

The following is an example of retrieving a list of s for records of Incident where is Network.

#!/usr/bin/perl -w
#use SOAP::Lite ( +trace => all, maptype => {} ); use SOAP::Lite;
sub
  SOAP::Transport::HTTP::Client::get_basic_credentials{return 'itil' => 'itil';}

my $soap = SOAP::Lite->proxy('http://<instance name>.service-now.com/incident.do?SOAP');

my $method = SOAP::Data->name('getKeys') -> attr({xmlns => 'http://www.service-now.com/'});

# get all incidents with category Network
my @params = ( SOAP::Data->name(category => 'Network'));

print $soap->call($method => @params)->result;

get

The following is an example of retrieving an Incident record using its sys_id value

#!/usr/bin/perl -w
use SOAP::Lite ( +trace => all, maptype => {} ); use SOAP::Lite;

sub
  SOAP::Transport::HTTP::Client::get_basic_credentials{return 'itil' => 'itil';}

my $soap = SOAP::Lite->proxy('http://<instance name>.service-now.com/incident.do?SOAP');

my $method = SOAP::Data->name('get') -> attr({xmlns => 'http://www.service-now.com/'});

# get incident by sys_id
my @params = ( SOAP::Data->name(sys_id => '9d385017c611228701d22104cc95c371'));

my %keyHash = %{ $soap->call($method => @params)->body->{getResponse};

# iterate through all fields and print them
foreach my $k (keys %keyHash)
  {print "$k=$keyHash{$k}\n";

getRecords

To query for an Incident using its incident number value:

#!/usr/bin/perl -w
use SOAP::Lite ( +trace => all, maptype => {} ); use SOAP::Lite;

sub
  SOAP::Transport::HTTP::Client::get_basic_credentials{return 'itil' => 'itil';}

my $soap = SOAP::Lite->proxy('http://<instance name>.service-now.com/incident.do?SOAP');

my $method = SOAP::Data->name('getRecords') -> attr({xmlns => 'http://www.service-now.com/'});

# get incident by number
my @params = ( SOAP::Data->name(number => 'INC10001'));

my %keyHash = %{ $soap->call($method => @params)->body->{getRecordsResponse}->{getRecordsResult};


getRecords (Returning Multiple Results)

The following is an example of retrieving and displaying an array of Incident records by querying all Incidents that have a category of 'Network'.

```perl
#!/usr/bin/perl -w
use SOAP::Lite ( +trace => all, maptype => {} );
sub SOAP::Transport::HTTP::Client::get_basic_credentials { return 'itil' => 'itil'; }
my $soap = SOAP::Lite->proxy('http://<instance name>.service-now.com/incident.do?SOAP');
my $method = SOAP::Data->name('getRecords')->attr({xmlns => 'http://www.service-now.com/' });
# get incident by sys_id
my @params = ( SOAP::Data->name(category => 'Network'));
my %keyHash = %{$soap->call($method => @params)->body->{'getRecordsResponse'}};
my $i = 0; my $size = @$keyHash{'getRecordsResult'}; for ($i = 0; $i < $size; $i++) {
  my %record = %{$keyHash{'getRecordsResult'}[$i]};
  print "------------------------------ $i -----------------------------
";
  foreach my $kk (keys %record) {
    print "$kk = $record{$kk} \n";
  }
}
```

deleteRecord

```perl
#!/usr/bin/perl -w
use SOAP::Lite ( +trace => all, maptype => {} );
sub SOAP::Transport::HTTP::Client::get_basic_credentials { return 'itil' => 'itil'; }
my $soap = SOAP::Lite->proxy('http://localhost:8080/glide/incident.do?SOAP');
my $method = SOAP::Data->name('deleteRecord')->attr({xmlns => 'http://www.service-now.com/' });
# delete incident by sys_id
my @params = ( SOAP::Data->name(sys_id => '46f67787a9fe198101e06dfcf3a78e99'));
my $result = $soap->call($method=>@params);
print_fault($result);
print_result($result);
sub print_result { my ($result) = @_;
  if ($result->body && $result->body->{'deleteRecordResponse'}) {
    my %keyHash = %{$result->body->{'deleteRecordResponse'}};
    foreach my $k (keys %keyHash) {
      print "name=$k   value=$keyHash{$k} \n";
    }
  }
  sub print_fault { my ($result) = @_;
  ```
Python web services client examples

Examples demonstrating an integration with a Python web services client.

Requirements

The following examples require the installation of the following Python modules:

- fpconst [http://pypi.python.org/pypi/fpconst/0.7.2](http://pypi.python.org/pypi/fpconst/0.7.2)
- PyXML [http://pyxml.sourceforge.net/topics/](http://pyxml.sourceforge.net/topics/)

insert

This is an example of inserting an incident.

```python
#!/usr/bin/python
from SOAPpy import SOAPProxy
import sys

def createincident (params_dict):
    # instance to send to
    instance = 'demo'

    # username/password
    username = 'itil'
    password = 'itil'

    # proxy - NOTE: ALWAYS use https://INSTANCE.service-now.com, not https://www.service-now.com/INSTANCE for web services URL from now on!
    proxy = 'https://%s:%s@%s.service-now.com/incident.do?SOAP' %
            (username , password , instance )
    namespace = 'http://www.service-now.com/
    server = SOAPProxy (proxy , namespace )

    # uncomment these for LOTS of debugging output
    #server.config.dumpHeadersIn = 1 #server.config.dumpHeadersOut = 1
    #server.config.dumpSOAPOut = 1 #server.config.dumpSOAPIn = 1

    response = server. insert (impact = int (params_dict['impact'] ) , urgency = int (params_dict['urgency'] ) , priority =
        int (params_dict['priority'] ) , category =params_dict['category'] , location =params_dict['location'] ,
        caller_id =params_dict['user'] , assignment_group =params_dict['assignment_group'] , assigned_to
        =params_dict['assigned_to'] , short_description =params_dict['short_description'] ,
        comments =params_dict['comments'] )

    return response
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
values = { 'impact': '1', 'urgency': '1', 'priority': '1', 'category': 'High', 'location': 'San Diego', 'user': 'fred.luddy@yourcompany.com', 'assignment_group': 'Technical Support', 'assigned_to': 'David Loo', 'short_description': 'An incident created using python, SOAPpy, and web services.', 'comments': 'This a test making an incident with python. Isn’t life wonderful?' }

new_incident_sysid = createincident (values )
print "Returned sysid: " + repr (new_incident_sysid )

getKeys

This is an example of executing getKeys on the demo instance using basic authentication.

#!/bin/env python

# use the SOAPpy module from SOAPpy import SOAPProxy

username , password , instance = 'admin' , 'admin' , 'demo'

server = SOAPProxy (proxy ,namespace )
response = server. getKeys (category = 'Network' )

print response. sys_id. split ( ',')

getRecords

In this example, we get an incident, querying for category == "Network" (with basic authentication).

#!/bin/env python

# use the SOAPpy module from SOAPpy import SOAPProxy

username , password , instance = 'admin' , 'admin' , 'demo'

server = SOAPProxy (proxy ,namespace )
response = server. getRecords (category = 'Network' )

for record in response:
    for item in record:
        print item

get

In this example, we get an incident record by sys_id (with basic authentication).

#!/bin/env python

# use the SOAPpy module from SOAPpy import SOAPProxy
Advanced

This is an example of advanced Python script that reads a log file for a keyword *invalid spi* and creates an ECC Queue record where the payload is set to an alert of XML format.

```python
#!/bin/env python

# kevin.pickard@service-now.com   2008.07.03  initial creation

from SOAPpy    import SOAPProxy
from xml.dom.minidom import getDOMImplementation
import sys , os , socket , pickle , re

# instance to send to
instance = 'demo'

# username/pass
username = 'admin'
password = 'admin'

# log file to watch
syslogfile = '/var/log/cisco.log.ksp'

# state file
statefile = '/tmp/syslog_ecc.state-test'

# ECC queue values
soapagent = 'SOAPpy'
ecctopic = 'PIX Error: '
ecccname = 'Invalid SPI: '
eccsource = 'Syslog'

# regex string to match
matchstring = 'invalid spi'

try:
    state = open (statefile , 'r' )
    lastbyte = pickle.load (state )
    state.close() except:
    lastbyte = 0

    #print 'DEBUG: lastbyte = '+str(lastbyte)

    try:
        log = open (syslogfile , 'ro' ) except:
            errortopic = 'Script Error'
            errorname = 'Unable to open log file '+'syslogfile+ '.
            errorpayload = 'This message was generated due to an error condition
                encountered in a script.  The name of the script is '+ os.path.basename (sys.argv[ 0 ] )+' on server '+ socket.gethostname ()+'
```
proxy = 'https://'+username+':' + password + '@' + instance + '.' + service-now.com/ecc_queue.do?SOAP'
namespace = 'http://www.service-now.com/
server = SOAPProxy (proxy, namespace)
server. config. dumpSOAPOut = 1
server. config. dumpSOAPIn = 1
response = server. insert (agent = soapagent, topic = errortopic, name = errorname, source = sys.argv [0], payload = errorpayload)
sys. exit (1)
if lastbyte != 0:
    try:
        log. seek (lastbyte) except IOError:
            pass

loglines = log. readlines()
lastbyte = log. tell()
log. close()

state = open (statefile, 'w') pickle. dump (lastbyte, state)
state. close()

# regex out the line
matchedlines = [] for line in loglines:
    if re. search (matchstring, line) != None:
        matchedlines. append (line)

# print 'DEBUG: len->loglines = '+str(len(loglines)) # print 'DEBUG: lastbyte = '+str(lastbyte) # print 'DEBUG: matchedlines = '+str(matchedlines)
if len(matchedlines) == 0:
    sys. exit(0)
proxy = 'https://'+username+':'+password+'@'+instance+'.'+service-now.com/ecc_queue.do?SOAP'
namespace = 'http://www.service-now.com/
server = SOAPProxy (proxy, namespace)
server. config. dumpSOAPOut = 1
#server. config. dumpSOAPIn = 1
entriestosend = {} for line in matchedlines:
    device = line. split() [3]
    sourceip = line. split() [-1]
    entriestosend [sourceip] = [device, line]

for key, value in entriestosend. items ():
    #impl= getDOMImplementation() #newdoc = impl. createDocument (None, "log_line", None) #top_element = newdoc. documentElement #text = newdoc. createTextNode (value[1]) #top_element. appendChild (text)

response = server. insert (agent = soapagent, topic = ecctopic+value[0], name = eccname+key, source = eccsource, payload = value[1])

Web services C Sharp .NET end to end tutorial

Examples demonstrating how to use .NET to consume a ServiceNow web service.

This tutorial will show you how to configure ServiceNow correctly to receive a web service request from your .NET client, as well as how to consume our web services using C# .NET.
Configure C# with .NET
Configure web services within ServiceNow.

1. To configure web services within ServiceNow, access the System Properties > Web Services module.

   This module displays the system properties that are specific to web services within your instance. For security reasons, you will want to make sure that you require basic authorization for incoming SOAP requests. This ensures that only authenticated users will be able to make any web services calls, whether it be via web service import sets or inserting/deleting/querying via direct web services.

2. This next step is very important if you are using .NET as a client to connect to ServiceNow. You must set the `elementFormDefault` property to false.

   This property defines how the WSDLs are qualified. Of course, if you do not consume our WSDL and just create the XML manually, then this property is irrelevant.


Call a web service in visual studio .NET
Call a web service using Visual Studio 2008.
In this example, we will be using Visual Studio 2008. First, create a new Windows Form Application for this example.

![New Project dialog box]

On the resulting form, we created a `richTextBox` (which we named `richTextBoxResult`) and a button (named `buttonResult`).

**Dot net project**

On the resulting form, we created a `richTextBox` (which we named 'richTextBoxResult') and a button (named 'buttonResult').
Dot net example form

Use a service reference in a C Sharp integration
Use a wizard to add a service reference for a C Sharp integration.

Go to the Solutions Explorer and select Service References > Add Service Reference. A wizard will appear asking for an address. Use: https://<instance name>.service-now.com/incident.do?WSDL. Accept the defaults for the rest of the wizard.

Open the app.config file and change the Security mode to "Transport" and the clientCredentialType and proxyCredentialType to "Basic"
Dot net app config

Use a web reference in a C Sharp integration
Use a wizard to add a web reference for a C Sharp integration.

Go to the Solutions Explorer and select Service References > Add Service Reference. A wizard will appear. At the bottom of the form, there is an Advanced button. Click on it and click on the Add Web Reference button at the bottom of the new wizard page. This will start the Web Reference wizard. For the URL, use: https://<instance name>.service-now.com/incident.do?WSDL and name the web reference, 'WebReference1'. Accept the defaults for the rest of the wizard.

C Sharp integration source code
After defining the source code, insert it.

Now we are ready to insert the code. Double-click on the Send Web Service button on your form to open the backend code to the form that has been created. Here is the code to insert a record into the demo instance and to read the response.

```csharp
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace ExampleWebServiceForWiki
{
    public partial class FormMain : Form
    {
        public FormMain()
        {
            InitializeComponent();
        }
        private void buttonSend_Click(object sender, EventArgs e)
        {
            /* SERVICE REFERENCE-SPECIFIC CODE
                ServiceReference1.ServiceNowSoapClient soapClient = new
                ServiceReference1.ServiceNowSoapClient();
                soapClient.ClientCredentials.UserName.UserName = "itil";
                soapClient.ClientCredentials.UserName.Password = "itil";
                ServiceReference1.insert insert = new
                ExampleWebServiceForWiki.ServiceReference1.insert();
                ServiceReference1.insertResponse response = new
                ExampleWebServiceForWiki.ServiceReference1.insertResponse();
                // END OF SERVICE REFERENCE CODE */
```
// WEB REFERENCE-SPECIFIC CODE
WebReference1.ServiceNow_incident soapClient = new
ExampleWebServiceForWiki.WebReference1.ServiceNow_incident();
System.Net.ICredentials cred = new
System.Net.NetworkCredential("itil", "itil");
soapClient.Credentials = cred;

WebReference1.insert insert = new WebReference1.insert();
WebReference1.insertResponse response = new
WebReference1.insertResponse();
// END OF WEB REFERENCE CODE

insert.category = "Category";
insert.comments = "Comments";
insert.short_description = "My short description";

try
{
    response = soapClient.insert(insert);
    this.richTextBoxResult.Text = "Incident Number: " + 
    response.number + "\n";
    this.richTextBoxResult.Text += "Sys_id: " + response.sys_id;
}
catch (Exception error)
{
    this.richTextBoxResult.Text = error.Message;
}

C Sharp integration results

If you have followed the tutorial correctly, you should receive the result whether you used a
Service Reference or a Web Reference.

Dot net tutorial results
Troubleshoot a null response in a C Sharp integration

Receiving a null response from ServiceNow’s web service.

If you are receiving a "null" response from your web service in your client code, then you may have missed the step in this tutorial for setting the elementFormDefault setting to "False". Here is a quick video tutorial that shows you how to do this.

Remember to recompile your code against the WSDL after you have changed this setting and saved it.

Retrieve a large number of records using SOAP

By default, a single SOAP request can retrieve a maximum of 250 records.

SOAP relies on Extensible Markup Language (XML) as its message format, and usually relies on other Application Layer protocols (most notably Remote Procedure Call (RPC) and HTTP) for message negotiation and transmission. SOAP can form the foundation layer of a web services protocol stack, providing a basic messaging framework upon which web services can be built. Because of the verbose XML format, SOAP can be considerably slower than other transport methods. Therefore, sending a large amount of data via SOAP is inefficient and is discouraged. Because of this, ServiceNow has imposed a hard-limit of 250 records that can be retrieved at any time in a single query. You may find that this limit poses some technological challenges for your integration design.

SOAP strategies

Retrieve the information that you need and make your integration more efficient.

Use filters to limit the number of results

One way to make your web service calls fit within the 250 record limit is to think about the design of your integrating application.

For example, let's assume that we are making an incident form in C# to show a user the incidents that are assigned to him.
Problematic query approach

The C# application makes a soap call to retrieve all of the incidents within ServiceNow. The application would then store the results locally in memory. When the user decides to view the incidents that are assigned to him, the application loops the internal array and displays the incidents that are assigned to the user.

A better query approach

The C# application makes a soap call to retrieve all of the incidents within ServiceNow that are assigned to the logged-in user. The results are stored locally in memory. When the user decides to view the incidents that are assigned to him, the application shows all the results to the user.

A performance-optimized query approach

The C# application makes no SOAP call initially. When a logged-in user decides to view the incidents that are assigned to him, the application presents him with the choice of viewing active, closed, etc. It gives him the ability to filter the results that he wants to see before the SOAP call is even made. Then, the user is only presented with the results that he wished to view.

*Use a local data store to pull data from*

If a large amount of data needs to be queried often, and the data does not need to be real-time, perform a sync of the ServiceNow table that you’re interested in with your integrating application’s data store.

Data push

- Using a scheduled job, ServiceNow can generate a csv/xml from a report and have it emailed to a specific location. The receiver might have a trigger to take the email attachment, parse it, and populate an internal table from which the application can communicate when the data is needed.
- Using a schedule job, ServiceNow can generate a csv/xml from a report and FTP it to an public FTP/FTPS location. The integrating product would consume this csv file on a regular basis and populate an internal table from which the application can communicate when the data is needed.

**Note:** Currently, the platform does not provide a method for extracting very large amounts of data and sending the output to an FTP server. However, a customization to perform that function is described at [here](#). The customization was developed for use in specific ServiceNow instances, and is not supported by ServiceNow Customer support. The method is provided as-is and should be tested thoroughly before implementation. Post all questions and comments regarding this customization to our community [forum](#).

Data pull

Using a cron job, a machine internal to your network can make a wget call to pull csv/xml data from any table within ServiceNow. The integrating product would consume this csv/xml file on a
regular basis and populate an internal table from which the application can communicate when the data is needed. Examples of the `wget` command that would be used:

- `wget --user=itil --password=itil --no-check-certificate https://<instance name>.service-now.com/incident_list.do?CSV`
- `wget --user=itil --password=itil --no-check-certificate https://<instance name>.service-now.com/incident_list.do?XML`

**Use Java/C#/PHP code to fetch the XML data using basic authentication**

If a local data store is not an option, another way to get the data is to call the CSV/XML processor directly and then parse the results.

Use the resulting data in a similar manner as you would a direct SOAP call. An example of this in PHP:

```php
<?php
// This example is in PHP

$user = "itil";
$pass = "itil";
$userPass = $user.':'.$pass;

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'https://<instance name>.service-now.com/incident_list.do?CSV');
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
curl_setopt($ch, CURLOPT_USERPWD, $userPass);

$data = curl_exec($ch);
$info = curl_getinfo($ch);

if ($output === false) {
    $output = "No cURL data returned for $addr [". $info['http_code'] ."]";
    if (curl_errno($ch))
        $output .= "\n" . curl_error($ch);
    print $output;
} else {
    echo $data;
}
curl_close($ch);
?>
```

**Excel web service**

You can export a list to an Excel file using an HTTP GET request.

**Excel web service request URLs**

Use one of several URL parameters to make a request to the Excel web service.
Excel URLs

<table>
<thead>
<tr>
<th>URL parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>?XLSX</td>
<td>Returns the requested list as a .XLSX file.</td>
</tr>
<tr>
<td>?EXCEL</td>
<td>Returns the requested list as a .XLS file. This parameter is equivalent to ?XLS.</td>
</tr>
<tr>
<td>?XLS</td>
<td>Returns the requested list as a .XLS file. This parameter is equivalent to ?EXCEL.</td>
</tr>
</tbody>
</table>

For example, to retrieve a list of incidents and save them to a .XLSX file, issue an HTTP GET to the following URL:

```
https://instance_name.service-now.com/incident.do?XLSX
```

The Content-Disposition header in the response displays the file name and extension of the resulting file. The file name is based on the table you are exporting from, such as incident.xlsx in this example.

**Excel web service parameters**

Use additional URL parameters to customize and filter the response.

#### Excel parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_query</td>
<td>Enter an encoded query string in this parameter. The parameter filters the data using the encoded query before returning the Excel file.</td>
<td>This example demonstrates how to filter an incident request to return only active incidents. <a href="https://instance_name.service-now.com/incident.do?EXCEL&amp;sysparm_query=active=true">https://instance_name.service-now.com/incident.do?EXCEL&amp;sysparm_query=active=true</a></td>
</tr>
<tr>
<td>sysparm_view</td>
<td>Enter the name of a list view to use to control which fields are returned.</td>
<td>This example demonstrates how to use the ESS view.</td>
</tr>
</tbody>
</table>

**CSV Web service**

An administrator can retrieve CSV data using an HTTP GET request.

Trigger a request using a csv URL parameter. For example, to retrieve a list of incidents and save them to a CSV file, issue an HTTP GET to the following URL: `https://instance_name.service-now.com/incident.do?CSV`

The Content-Disposition HTTP header in the response indicates the file name and extension of the extract; “incident.csv” in the example.
CSV Web service parameters
The following URL parameters customize and filter the CSV content that is returned by an HTTP GET request.

sysparm_query
Enter an encoded query string in this parameter. Use the parameter to filter the data using the encoded query before returning the CSV content. The following request filters the list to return only active incidents:

https://instance_name.service-now.com/incident.do?CSV&sysparm_query=active=true

sysparm_view
The value indicates which view to use to limit the field values that are returned. For example, to use the "ess" view:

https://instance_name.service-now.com/incident.do?CSV&sysparm_view=ess

Adding data to tables by importing a CSV file
You can create new data by importing a CSV file directly to a table.

The CSV file headers must match the field columns in the targeted table.

JSONv2 Web Service
The ability to describe sets of data in JSON format is a natural extension to the JavaScript language.

ServiceNow supports a web service interface that operates on the JSON object as the data input and output format.

The JSONv2 web service is provided by a platform-level processor similar to the services for SOAP, WSDL, CSV, Excel, and XML. Like those services, the JSON web service is triggered by a standalone JSONv2 URL parameter. For example:

https://<instance name>.service-now.com/mytable.do?JSONv2

Having the JSON object available as a data format for web services means that you can create (insert), update, and query any data in the system using the JSON object format, and get results in the JSON object format.

Security

Like all other HTTP-based web services available on the platform, the JSONv2 web service is required to authenticate using basic authentication by default. The user ID that is used for authentication is subjected to access control in the same way as an interactive user.

JSON object format

The JSON object is built in two structures.

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

In its simplest form, a JSON object is just a comma delimited set of name/value pairs. For example:

```json
{"name one":"value one","name two":"value two"}
```

The following is a sample of a single record array of incidents in JSON:

```json
{"records":
[{"closed_by":"",
"__status": "success",
"category": "inquiry",
"escalation": "0",
"state": "1",
"location": "",
"reassignment_count": "0",
"time_worked": "",
"order": "0",
"due_date": "",
"number": "INC0010180",
"upon_approval": "proceed",
"sla_due": "2010-03-04 22:51:49",
"follow_up": "",
"notify": "1",
"business_stc": "0",
"caused_by": "",
"rejection_goto": "",
"assignment_group": "d625dccec0a8016700a222a0f7900d06",
"incident_state": "1",
"opened_at": "2010-02-23 22:51:49",
"wf_activity": "",
"calendar_duration": "",
"group_list": "",
"caller_id": "",
"comments": "",
"priority": "3",
"sys_id": "fd0774860a0a0b380061bab9094733ad",
"sys_updated_by": "itil",
"variables": "",
"delivery_task": "",
"sys_updated_on": "2010-02-23 22:51:49",
"parent": "",
"active": "true",
"opened_by": "681b365ec0a80164000fb058754a0cd",
"expected_start": "",
"sys_meta": "System meta data",
"watch_list": "",
"company": "",
"upon_reject": "cancel",
"work_notes": "",
"sys_created_by": "itil",
"cmdb_ci": "",
"approval_set": "",
"user_input": "",
"sys_created_on": "2010-02-23 22:51:49",
"contact_type": "phone",
"rfc": "",
"approval_history": "",
"activity_due": "",
"severity": "3",
"subcategory": "",
"work_end": ""}]
```
The following is a record array of incident responses with an error.

```json
{  "records": [    {      "__error": {        "message": "Invalid Insert into: incident",        "reason": "Data Policy Exception: Short description is mandatory"      },      "__status": "failure",      "active": "true",      "activity_due": "",      "approval": "not requested",      "approval_history": "",      "approval_set": "",      "assigned_to": "",      "assignment_group": "d625dccec0a8016700a222a0f7900d06",      "business_duration": "",      "business_stc": "",      "calendar_duration": "",      "calendar_stc": "",      "caller_id": "",      "category": "inquiry",      "caused_by": "",      "child_incidents": "0",      "close_code": "",      "close_notes": "",      "closed_at": "",      "closed_by": "",      "cmdb_ci": "",      "comments": "",      "comments_and_work_notes": "",      "company": "",      "contact_type": "phone",      "correlation_display": "",      "correlation_id": "",      "delivery_plan": ""    }  ]}
```
"delivery_task": "",
"description": "",
"due_date": "",
"escalation": "0",
"expected_start": "",
"follow_up": "",
"group_list": "",
"impact": "3",
"incident_state": "1",
"knowledge": "false",
"location": "",
"made_sla": "true",
"notify": "1",
"number": "INC0010001",
"opened_at": "2013-07-23 18:01:17",
"opened_by": "6816f79cc0a8016401c5a33be04be441",
"order": "",
"parent": "",
"parent_incident": "",
"priority": "5",
"problem_id": "",
"reassignment_count": "0",
"reopen_count": "0",
"resolved_at": "",
"resolved_by": "",
"rfc": "",
"severity": "3",
"short_description": "",
"skills": "",
"sla_due": "",
"state": "1",
"subcategory": "",
"sys_class_name": "incident",
"sys_created_by": "admin",
"sys_created_on": "2013-07-23 18:01:17",
"sys_domain": "global",
"sys_id": "a96479343cb60100a92ec9a477ba9e45",
"sys_mod_count": "0",
"sys_updated_by": "admin",
"sys_updated_on": "2013-07-23 18:01:17",
"time_worked": "",
"upon_approval": "proceed",
"upon_reject": "cancel",
"urgency": "3",
"user_input": "",
"watch_list": "",
"work_end": "",
"work_notes": "",
"work_notes_list": "",
"work_start": ""
]
}

JSON response status

JSONv2 requests may return one of several response statuses.
**JSON Success Response**

Each JSON success response includes a record array containing the records retrieved by the given action. Each JSON object contains one or more metadata elements, prefixed with __, regarding the status for the action on each record, as illustrated in the previous examples. The JSON success responses use the following syntax:

```
__status
"__status": "<value>"
```

where <value> is success or failure.

**JSON Failure Response**

When the _status element returns failure, the _error element is added to identify the error and reason.

```
"__error": { "message": "<error value>", "reason": "<reason value> "}
```

where <error value> is the error message text and <reason value> is the reason the error was triggered.

The JSON error response contains only the error and reason elements. Generally, this indicates that the whole JSON operation failed and no records can be processed.

For example:

```
{"__error":"Cannot update with empty sysparm_query","reason":null}
```

**Setting the number of rows returned**

The following system property controls how many rows JSON returns with each query.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>glide.processor.json.row_limit</td>
<td>Specify the maximum number of rows a JSON query returns.</td>
</tr>
<tr>
<td></td>
<td>• Type: integer</td>
</tr>
<tr>
<td></td>
<td>• Default value: 10,000</td>
</tr>
<tr>
<td></td>
<td>• Location: Add to the System Properties [sys_properties] table</td>
</tr>
</tbody>
</table>

**Requiring basic authentication for incoming JSONv2 requests**

The following system property controls whether basic authentication is required for incoming JSONv2 requests.
Requiring Basic Authentication for Incoming JSONv2 Requests

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>glide.basicauth.required.jsonv2</td>
<td>Enables (true) or disables (false) requiring basic authentication for incoming JSONv2 requests.</td>
</tr>
<tr>
<td></td>
<td>- Type: true</td>
</tr>
<tr>
<td></td>
<td>- Default value: true</td>
</tr>
<tr>
<td></td>
<td>- Location: Add to the System Properties (sys_properties) table</td>
</tr>
</tbody>
</table>

Action parameters

Action parameters are separate and different from data parameters because they specify the action to take when the JSON object parameter is part of an HTTP GET or POST request.

The parameters can also be specified as a field in the supplied JSON object. They have the effect of triggering an action in the case of `sysparm_action`, or filtering the results of an update or query in the case of `sysparm_query`.

sysparm_action

The following are the valid values for `sysparm_action` and the corresponding action triggered by the API.

**Data Retrieval**

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getKeys</td>
<td>Query the targeted table using an encoded query string and return a comma delimited list of <code>sys_id</code> values.</td>
</tr>
<tr>
<td>getRecords</td>
<td>Query the targeted table using an encoded query string and return all matching records and their fields.</td>
</tr>
<tr>
<td>get</td>
<td>Query a single record from the targeted table by specifying the <code>sys_id</code> in the <code>sysparm_sys_id</code> URL parameter, and return the record and its fields.</td>
</tr>
</tbody>
</table>

**Data Modification**

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert</td>
<td>Create one or more new records for the table targeted in the URL.</td>
</tr>
<tr>
<td>insertMultiple</td>
<td>Create multiple new records for the table targeted in the URL.</td>
</tr>
<tr>
<td>update</td>
<td>Update existing records in the targeted table in the URL, filtered by an encoded query string.</td>
</tr>
<tr>
<td>Method Summary</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>deleteRecord</td>
<td>Delete a record from the table targeted in the URL by specifying its sys_id in the sysparm_sys_id URL parameter.</td>
</tr>
<tr>
<td>deleteMultiple</td>
<td>Delete multiple records from the table targeted in the URL, filtered by an encoded query string.</td>
</tr>
</tbody>
</table>

**sysparm_query**

Specify an encoded query string to be used in get, getRecords, update or deleteMultiple sysparm_action value.

**sysparm_view**

Specify a form view to customize the return values for get and getRecords function calls. When using a view, the query returns only the fields defined in the view, including referenced values. If there is no view name, or if the view name is not valid, then the query returns all field names that are marked active in the dictionary.

**sysparm_sys_id**

Specify a target sys_id during a get or delete function call (sysparm_action value).

**sysparm_record_count**

Specify an integer value to limit the number of records retrieved for this request. Note that this value is capped by the glide.processor.json.row_limit system property.

**displayvalue**

Get the display value of a reference field, if any are in the record. For example, the Incident record can have an assigned_to field that is a reference to a user record. Instead of sending the sys_id of the user record, the user name is sent.

The displayvalue parameter can have three values: **true**, **false**, or **all**.

- **true**: All the references fields show the display value instead of sys_id.
- **false** (default): All reference fields show sys_ids.
- **all**: The display value and the sys_id are shown. For example, the assignedto field in the Incident record is sent back as assigned_to:1234556, dv_assigned_to:Fred Luddy.

**displayvariables**

Set this boolean value to **true** during a get or getRecords function call to retrieve all variables attached to this record.
JSON Data Retrieval API

Query for data by issuing an HTTPS GET request to the instance.

By default, a GET request is interpreted as a get function if a `sysparm_sys_id` parameter is present. Otherwise, it is interpreted as a getRecords function. You can also specify a URL parameter `sysparm_action=get`. Query responses are always encapsulated by a records hash of records, where each individual record's values are themselves hashed by field name.

Return Display Value for Reference Variables

When you are getting a record from a get or getRecords function, all the fields associated with that record are returned. The fields are often reference fields that contain a `sys_id` for another table. The base system behavior is to return the `sys_id` value for those fields. To have the display value for the field returned, use one of these options:

- Add the property `glide.json.return_displayValue` to the system properties, and every JSON request will return a display value for a reference field.
- Add the parameter `displayvalue=true` to the JSON request URL and JSON requests with that parameter will return a display value instead of the `sys_id` for a reference field. The JSON URL would look like this:

```url
https://<instance name>.service-now.com/incident.do?JSON&sysparm_action=getRecords&sysparm_query=active=true^category=hardware&displayvalue=true
```

- Add the parameter `displayvalue=all` to the JSON request URL and JSON requests with that parameter return a display value and the `sys_id` for a reference field. The response element name for the display value field will be prefixed with `dv_`, for example `dv_caller_id`.

Return Display Variables

Use the `displayvariables` parameter to return an array of variables associated with a service catalog item record when using a get or getRecords function. The variables are expressed hierarchically. A variable container has the container's variables in its children field.

To get display variables, add the parameter `displayvariables=true` to the JSON request URL. The JSON request URL would look like:

```url
https://<instance name>.service-now.com/sc_req_item.do?v2&sysparm_action=getRecords&sysparm_query=active=true^short_description=Laptop%20preconfigured%20for%20developers&displayvariables=true
```

White space has been added to the following example for clarity:

```
```
Control the order of records

You can control the order that records appear in the JSON response. To set an order, use the ORDERBY or ORDERBYDESC clauses in the URL encoded query. For example,

```
sysparm_query=active=true^ORDERBYnumber^ORDERBYDESCcategory
```

filters all active records and orders the results in ascending order by number first, and then in descending order by category. For more information, see Encoded query strings.

def getKeys

Get the sys_id of multiple records by specifying an encoded query string in the sysparm_query parameter.

```
https://<instance name>.service-now.com/incident.do?
JSONv2&sysparm_action=getKeys&sysparm_query=active=true^category=hardware
```

def get

Get a record directly by specifying the sys_id in a sysparm_sys_id parameter.

```
https://<instance name>.service-now.com/incident.do?
JSONv2&sysparm_sys_id=9d385017c611228701d22104cc95c371
```

Optionally, you may also specify the sysparm_action parameter:

```
https://<instance name>.service-now.com/incident.do?
JSONv2&sysparm_action=get&sysparm_sys_id=9d385017c611228701d22104cc95c371
```

def getRecords

Get all records by specifying an encoded query string in the sysparm_query parameter.

```
https://<instance name>.service-now.com/incident.do?
JSONv2&sysparm_action=getRecords&sysparm_query=active=true^category=hardware
```

JSON Data Modification API

Modify data using the JSON web service by sending an HTTPS POST request to the instance. The HTTP POST must contain a sysparm_action parameter to indicate the type of action to be performed, with the incoming JSON object post in the body.
insert

Create a new record in ServiceNow. The JSON object has to be POSTed as the body (content-type is usually application/json, although not enforced). The response from the record creation is a JSON object of the incident that was created.

For example, posting the following JSON object:

```
{"short_description":"this is a test","priority":"1"}
```

to the following URL:

```
https://your_instance.service-now.com/incident.do?JSONv2&sysparm_action=insert
```

creates an incident.

Optionally, you may also specify the sysparm_action in the JSON object. The parameter inside the JSON object takes precedence over the URL parameter. For example:

```
{"sysparm_action":"insert","short_description":"this is a test","priority":"1"}
```

insertMultiple

To create multiple new records in ServiceNow, the input JSON object for the insert function must be an array. The response from the record creation is a JSON object of the incidents that were created. For example, the following JSON object:

```
{
"records" : [ { "short_description" : "this was inserted with python using JSON 1", "priority" : "1 - Critical", "impact" : "1", "caller_id" : "Fred Luddy" }, { "short_description" : "this was inserted with python using JSON 2", "priority" : "1 - Critical", "impact" : "1", "caller_id" : "Fred Luddy" } ]
}
```

posted to one the following URLs:

```
https://<instance name>.service-now.com/incident.do?JSONv2&sysparm_action=insert
https://<instance name>.service-now.com/incident.do?JSONv2&sysparm_action=insertMultiple
```

creates two incidents. Note the fields described as an array value for the records field.

update

Update a record or a list of records filtered by an encoded query string specified by the sysparm_query URL parameter. The JSON object has to be posted as the body (content-type is usually application/json, although not enforced). The response from the record creation is an array of JSON objects representing the records that were updated.
For example, posting the following JSON object:

```json
{"short_description":"this was updated with python", "priority": "3", "impact":"1"}
```

to the following URL:

```plaintext
https://instance_name.service-now.com/incident.do?JSONv2&sysparm_query=priority=3&sysparm_action=update
```

updates all incidents with priority 3, and sets the values specified by the JSON object.

**deleteRecord**

Delete a single record from the targeted table, identified by a `sysparm_sys_id` parameter. The parameter may be encoded in the input JSON object or given as a URL parameter.

For example, posting:

```json
{"sysparm_sys_id":"fd4001f80a0a0b380032ffa2b749927b"}
```

to the following URL:

```plaintext
http://instance_name.service-now.com/incident.do?JSONv2&sysparm_action=deleteRecord
```

deletes the incident record identified by the sys_id fd4001f80a0a0b380032ffa2b749927b.

**deleteMultiple**

Delete multiple records from the targeted table, filtered by an encoded query string specified in the `sysparm_query` URL parameter. The filter may also be encoded in the input JSON object.

For example, posting:

```json
{"sysparm_query":"short_description=this was updated with python"}
```

to the following URL:

```plaintext
http://instance_name.service-now.com/incident.do?JSONv2&sysparm_action=deleteMultiple
```

deletes all incident records where the `short_description` field contains the value “this was updated with python”.

**PDF web service**

The Now Platform supports programmatic retrieval of PDF data through an HTTP GET request. The request is triggered when you use the `PDF` URL parameter.

For example, to retrieve a list of "incidents" and save to a PDF file, issue an HTTP GET to the URL:

```plaintext
https://instance_name.service-now.com/incident_list.do?PDF
```

The “content-disposition” HTTP header in the response indicates the file name and extension of the extract. In the example, it will be “incident.pdf”
PDF web service parameters

The following URL parameters customize and filter the PDF content that is returned by an HTTP GET request.

sysparm_query

The value of the sysparm_query parameter is an encoded query string. Use the parameter to filter the data using the encoded query before returning the PDF content. The following request filters the list to return only active incidents:

https://instance_name.service-now.com/incident.do?PDF&sysparm_query=active=true

sysparm_view

The value of the sysparm_view parameter specifies which list view to use to limit the field values that are returned. For example, to use the “ess” view, use:

https://instance_name.service-now.com/incident.do?PDF&sysparm_view=ess

To generate a list, use:

https://instance_name.service-now.com/incident_list.do?PDF&sysparm_view=ess

XML web service

The Now Platform supports programmatic retrieval of XML data through a URL query (HTTP GET request).

The request is triggered by use of a URL parameter. The following XML-based export formats from a URL query are supported.

<table>
<thead>
<tr>
<th>Export Processor</th>
<th>URL Syntax</th>
<th>Example URL</th>
</tr>
</thead>
</table>

An example XML document retrieved using a URL with the following pattern https://instancename.service-now.com/change_request.do?XML is displayed below. Access control is protected using Basic Authentication.

```xml
<xml>
  <change_request>
    <description>Please install new Cat. 6500 in Data center 01</description>
  </change_request>
</xml>
```
<category>Hardware</category>
<due_date/>
<scope>3</scope>
<comments/>
<knowledge>false</knowledge>
<active>true</active>
<phase>requested</phase>
<justification/>
<cab_date/>
<review_date/>
.sys_created_on>2009-04-14 23:14:14</sys_created_on>
<correlation_id/>
<follow_up/>
<sys_domain>global</sys_domain>
<close_notes/>
<urgency>3</urgency>
<change_plan/>
<business_duration/>
<sys_id>46cb2f54a9fe198101cf6814a2754606</sys_id>
<state>1</state>
<reason/>
<closed_at/>
<closed_by/>
.sys_updated_on>2009-12-21 23:47:15</sys_updated_on>
<contact_type>phone</contact_type>
<group_list/>
<risk>3</risk>
<assignment_group/>
<sys_updated_by>pat.casey</sys_updated_by>
<production_system>false</production_system>
<approval_history/>
<sys_created_by>glide.maint</sys_created_by>
<end_date>2009-12-21 00:30:00</end_date>
<order/>
<variables/>
<calendar_duration/>
<work_start/>
<backout_plan/>
<start_date>2009-12-20 19:30:00</start_date>
<correlation_display/>
<made_sla>false</made_sla>
<location/>
<test_plan/>
<approval_set/>
<sys_class_name>change_request</sys_class_name>
<sys_mod_count>6</sys_mod_count>
<review_comments/>
<phase_state>open</phase_state>
<time_worked/>
<type/>
<implementation_plan/>
<priority>1</priority>
<user_input/>
<work_end/>
<rejection_goto/>
<upon_reject/>
requested_by_date>2009-04-08 00:00:00</requested_by_date>
<delivery_plan/>
<parent/>
<delivery_task/>
<short_description>Install new Cisco</short_description>
<assigned_to>681b365ec0a80164000fb0b5854a0cd</assigned_to>
<approval>requested</approval>
<escalation>0</escalation>
XML web service parameters

ServiceNow Technical Supports the URL parameters, sysparm_query and useUnloadFormat, to customize and filter the response you get.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysparm_query</td>
<td>The value for this URL parameter is that of an encoded query string and when used, will filter the data using the encoded query before returning the XML file. The following request will filter the list to only return &quot;incidents&quot; that are active: <a href="https://instance_name.service-now.com/incident.do?XML&amp;sysparm_query=active=true">https://instance_name.service-now.com/incident.do?XML&amp;sysparm_query=active=true</a></td>
</tr>
<tr>
<td>useUnloadFormat</td>
<td>This parameter indicates that the XML format that is returned to be of an unload format. The unload format is the same format you get when, from a list in the UI, you select Export &gt; XML. The unload-formatted XML can be imported back into the instance by administrators of your system. To enable the unload format from a URL, use the useUnloadFormat=true URL parameter. For example: <a href="https://instance_name.service-now.com/incident.do?XML&amp;useunloadFormat=true">https://instance_name.service-now.com/incident.do?XML&amp;useunloadFormat=true</a></td>
</tr>
</tbody>
</table>

Using the URL parameter produces an XML format that looks like:

```xml
<unload unload_date="2010-03-12 17:48:30">
  <incident action="INSERT_OR_UPDATE">
    <description/>
    <category>inquiry</category>
    <due_date/>
  </incident>
</unload>
```
RSS web service

RSS (Rich Site Summary) is a format for delivering web-based information that changes regularly.

RSS feed generator

ServiceNow supports the dynamic generation of RSS feeds.

Much like our Web Services implementation, the retrieval of an RSS feed representation of information is simply done by specifying an RSS parameter at the end of the URL to a table list. For example, the following will return a list of all incidents in RSS 2.0 format:

Adding a Query

To associate a query to the list so that a filtered list is returned, use the `sysparm_query` parameter. For example, the following will return a list of all incidents where the priority field is 1 (Critical):

```
https://<instance name>.service-now.com/incident.do?
sysparm_query=priority=1&RSS
```
If you have a multi part query then you would separate the parts with the ^ character. For example to get all priority 1 incidents with a category of software you would:

```
https://<instance name>.service-now.com/incident.do?
sysparm_query=priority=1^category=software&RSS
```

If you want to query on a field that is a reference to another file then you need to use javascript to resolve the reference to the other file. For example, the assigned_to field in incident is a reference to a user record. If you wanted to find all the incidents assigned to 'ITIL User' then you would do the following:

```
https://<instance name>.service-now.com/incident.do?
sysparm_query=assigned_to=javascript:GetIDValue('sys_user','ITIL User')&RSS
```

**Note:** You can in most cases simply append "&RSS" to a URL that you generate in the U.I. or that of your favorite module. The easiest way to get the URL is to simply click the last breadcrumb from the list view. After appending "&RSS" then you can use this URL in your RSS feed reader.

**Limiting results with a view**

The description element in the returned RSS xml is constructed using the view as specified in the URL, when no view is specified, the default no-name view is used.

To change this format, specify the `sysparm_view` parameter on the URL. For example, the following request will return the incidents list. However the result will be restricted to only the fields available in the `ess` view:

```
https://<instance name>.service-now.com/incident.do?
sysparm_query=priority=1&sysparm_view=ess&RSS
```
Additionally, the RSS item title can be modified using the `sysparm_title_view` URL parameter. When specified, the item title will be constructed using the fields specified in the view. For example:

```
https://<instance name>.service-now.com/incident.do?
sysparm_query=priority=1&sysparm_view=ess&sysparm_title_view=rss_title&RSS
```

### Formatting results

The description element in the returned RSS xml can be formatted by setting the URL parameter `sysparm_format=true` and specifying the format string in the property `glide.rss.description_format`. By default, when the URL parameter is present, the description element will be formatted to contain the field label and value using the following format string:

```
<b>{1}</b>: {2}<br/>
```

- `{0}` - field name
- `{1}` - field label
- `{2}` - field value

This default format string can be overridden using the property `glide.rss.description_format`. An example of the formatted RSS feed can be seen in the following screen capture from Firefox:

```
Problem (All)
Problem (All)

PRB00001
Sun, Aug 10, 2008 3:39 PM

Number: PRB00001
Escalation: Moderate
Short description: Windows xp SP2 causing errors in Enterprise
Problem state: Pending Change
RFC: CHG00003

PRB00008
Sun, Aug 10, 2008 3:40 PM

Number: PRB00008
Escalation: Normal
Short description: Hang when trying to print VISIO document
Problem state: Open
RFC:
```

### RSS basic authentication

To enforce basic authentication on each request for an RSS feed, set the property `glide.basicauth.required.rss` to `true`. 
RSS request would have to contain the Authorization header as specified in the Authentication protocol. Because the request is non-interactive, we always require the **Authorization** header during a request.

**Note:** If you plan to disable RSS basic authentication, make sure that tables in the platform have the right ACL entries to protect from unauthorized access.

To specify basic authentication on the URL, put the username and password pair separated by a colon in front of the server name before an @ character. For example, to submit the demo credentials for the ITIL user, use the following URL:

```
https://itil:itil@<instance name>.service-now.com/incident.do?RSS
```

Some older browsers, such as Microsoft IE 7 do not support direct URL authentication. If the site uses basic authentication, Internet Explorer automatically prompts users for a user name and a password. In some cases, users can click the Remember my password box in the prompt to save their credentials for later visits to that site.

**RSS title override**

You may optionally override the automatically generated title of the RSS feed by added the `sysparm_title` parameter to the request URL.

For example, you can specify the title *Priority One Incidents* using the following request URL.

```
https://<instance name>.service-now.com/incident.do?
sysparm_query=priority=1&sysparm_view=ess&RSS&sysparm_title=Priority%20One%20Incidents
```

This will produce results as follows:
RSS Out

RSS feed reader

Create a scrolling RSS feed reader using a UI page.

You must create a feed parser using an RSS API, such as the Google Feed API.

Developer’s guide: http://code.google.com/apis/ajaxfeeds/documentation/

Scrollable areas

A scrollable area is a div where contents scroll from the bottom up over time.

You can scroll any HTML content, and anything inside the scroll is operational HTML with functioning links and images.

To make a scrollable areas, wrap the scrolling content inside of a scrollable_area tag, likely in a UI Page:

```html
<g:scrollable_area height="100px">
  <g2:evaluate var="jvar_temp" expression="var kb = new GlideRecord('kb_knowledge');"/>
  <g:inline template="kb_section.xml"/>
</g:scrollable_area>
```
The system will then create a 100 pixel high div and the contents will automatically scroll from bottom to top. If you have a 1000 pixel high block of text, for example, you'll see the top 100 pixels and then pixels 2-101, then 3-102, etc. Once it reaches the top it'll wrap back around to the bottom.

This sample code will create a scroller with a list of priority 1 incidents.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<j:jelly trim="false" xmlns:j="jelly:core" xmlns:g="glide" xmlns:j2="null"
xmlns:g2="null">
  <g2:evaluate var="jvar_inc">
    var inc = new GlideRecord('incident');
    inc.addActiveQuery();
    inc.addQuery('priority',1);
    inc.query();
  </g2:evaluate>

  <g2:scrollable_area height="100px">
    <table border="0" cellspacing="2" cellpadding="0" width="100%">
      <j2:while test="![inc.next()]">
        <j2:set var="jvar_inc_link" value="incident.do?sys_id=
        ![inc.sys_id]"/>
        <j2:set var="jvar_inc_list_link" value="incident_list.do?sysparm_query=active=true"/>
        <tr>
          <td>
            <a href="![jvar_inc_link]">
              <img src="images/services.png" style="padding-right:10px"></img>
            </a>
            <a href="![jvar_inc_link]" style="padding-right:10px; color:blue">![inc.number]</a>
          </td>
          <td>![inc.short_description]</td>
        </tr>
      </j2:while>
      <tr>
        <td align="center" colspan="2">
          <a href="![jvar_inc_list_link]" style="color:blue">View all active incidents</a>
        </td>
      </tr>
    </table>
  </g2:scrollable_area>
</j:jelly>
```

### Add scrolling elements in forms
You can add scrolling areas to forms as well as UI pages.

1. Create a UI Macro with the script.
2. Create a Formatter to reference the script.

### Priority 1 incidents example
This example scrolling element demonstrates how to create a UI macro to a scrolling list of priority 1 incidents.

Use the following example code:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<j:jelly trim="false" xmlns:j="jelly:core" xmlns:g="glide" xmlns:j2="null"
xmlns:g2="null">
  <g2:evaluate var="jvar_inc">
```

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
var inc = new GlideRecord('incident');
inc.addActiveQuery();
inc.addQuery('priority',1);
inc.query();
</g2:evaluate>

<div style="background-color:DDDDDD; padding-left:10px; line-height:19px;
border:2px white solid" width="100%" nowrap="true">
Priority 1 Incidents:
<input id="make_spacing_ok" style="visibility:hidden;width:0px;"
title=""/>
</div>
</div>
<g2:scrollable_area height="100px" width="100%">
<j2:while test="$[inc.next()]">
<j2:set var="jvar_inc_link" value="incident.do?sys_id=
$[inc.sys_id]"/>
<j2:set var="jvar_inc_list_link" value="incident_list.do?
sysparm_query=active=true"/>
<span style="line-height: 10px; padding-left:10px">
<a href="$[jvar_inc_link]">
<IMG SRC="images/services.png" style="padding-right:10px"></IMG>
</a>
<a href="$[jvar_inc_link]" style="padding-right:10px; color:blue">
$[inc.number]</a>
$[inc.short_description]
<br style="line-height:5px"/>
</j2:while>
<span>
<a href="$[jvar_inc_list_link]" style="color:blue; padding-left:10px">View all active incidents</a>
</span>
</g2:scrollable_area>
</j:jelly>

Navigate to **System UI > Formatters** and create a Formatter that refers to the UI Macro above.

**Add a scrolling news panel to your homepage**

To add a scrolling news panel to your homepage:

1. Navigate to a homepage.
2. Click **Add Content** to get a list of possible content types.
3. Click **Scrollers** in the leftmost box of the add content dialog.
4. Click the news scroller.
5. Click **Add** to add it on your page.
News in the scroller

The news scroller gets its news list by going to the Knowledge Base and querying the short descriptions of any items there with a topic of “News”.

To add news, go to the Knowledge Base and add new entries with a topic of "News".

Style Control

If you want to make a headline standout in the news scroller, use HTML tags on the short description. For example, enter <b>My Bold Test</b> and My Bold Test will appear in the scroller.

Note: The property glide.ui.escape_text must be false to use HTML formatting.

Scrolling News Panel Notes and Limitations

Scrollable news panels offer a way for you to display high impact news items on your users home pages in an eye catching scrolling news display.

- Scrolling is currently always from the bottom to the top.
- Scrolling is 1 pixel a at a time every 100 ms.
- You can have as many scrollable areas on a screen as you want (they get system generated unique names).
- Works in Internet Explorer and Firefox.
- If the object you are scrolling is shorter than the scrollable area, it still scrolls.
- If you mouse over a scrollable area, it stops scrolling (so you can click something without chasing it as it scrolls).

RSS feed reader example

An example of how to set up an RSS feed reader using an RSS feed.

For an example RSS feed reader, refer to the Now Community.

Web services security

Web services security is enforced using a combination of basic authentication challenge/response for the HTTP protocol and system-level access control using the Contextual Security Manager.

To enforce basic authentication on each Web Service request, each request must contain the Authorization header as specified in the Basic Authentication protocol. Because the request is non-interactive, the Authorization header is required in a request.

There is an added advantage when you supply basic authentication information whether or not it is required: the data that is created or updated as a result of the Web service invocation is done on behalf of the user supplied in the basic authentication credentials. For example, when creating an Incident record, the journal fields will contain the user ID of basic authenticated user, instead of the default “Guest” user.

REST & SOAP API analytics

REST & SOAP API analytics allows you to track and analyze web service API usage.

Use analytics to help answer questions such as
• Which APIs are used most?
• Which API versions are being used? Can I deprecate older versions?
• Which API methods are being used?
• What resources are being accessed?
• Who is using each API and resource?

The instance tracks analytics for all inbound web services including platform web services such as the REST Table API or the SOAP API, and custom web services such as Scripted REST APIs and Scripted SOAP web services. Analytics are aggregated by each resource and HTTP action combination.

Outbound web services such as REST Messages are not tracked.

**Whitelist an API to collect analytics**

Select which APIs to collect analytics for, and select if requestor information should be collected for each API.

**Note:** API analytics is not available for certain APIs used for internal platform functionality, such as the UI and Mobileapp APIs.

Role required: api_analytics_read or admin

When a new API is added, a whitelist record is created automatically. You can modify or create new whitelist records to manually configure which APIs and requestor information are logged.

Navigate to System Web Services > Whitelist and create a new record.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API name</td>
<td>For REST APIs, enter the API name. This name should contain only the namespace and ID of the API. Do not include any request parameters in the API name. For example, for the Table API endpoints api/now/table/incident and api/now/table/problem, the namespace and ID are now/table. SOAP APIs do not support separate logging configurations for different APIs. You can configure logging for all SOAP APIs by modifying the SOAP APIs whitelist record.</td>
</tr>
<tr>
<td>Collect API stats</td>
<td>Select this check box to track analytics for the specified API.</td>
</tr>
<tr>
<td>Collect API requestor stats</td>
<td>Select this check box to track which users make requests to this API. This option is available only if Collect API stats is selected.</td>
</tr>
</tbody>
</table>

**REST & SOAP API analytics dashboards**

You can view API analytics for the entire instance or for specific APIs using the included dashboards.
REST & SOAP API Analytics includes dashboards that present the overall analytics, analytics per API, and analytics per requesting user.

To access API Analytics dashboards, navigate to System Web Services and select Usage Overview, Usage by API, or Usage by Requestor. Users with the api_analytics_read or admin roles can view these dashboards.

**Note:** The Usage by Requestor dashboard uses the Responsive Canvas feature available with Istanbul. For the best experience on upgraded instances, you may need to enable responsive dashboards.

The following dashboards are available:

<table>
<thead>
<tr>
<th>Dashboard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage Overview</td>
<td>Provides general usage statistics for all REST and SOAP APIs.</td>
</tr>
<tr>
<td>Usage by Web API</td>
<td>Provides detailed usage statistics for each API.</td>
</tr>
<tr>
<td>Usage by Requestor</td>
<td>Provides detailed usage statistics for each requesting user. You can additionally filter by API to view detailed stats for a specific user and API combination. If the selected user has not made requests to the selected API, no data is shown for that combination.</td>
</tr>
</tbody>
</table>

**Note:** To view API stats for all users, do not clear the user breakdown selection. Instead, use the Usage by Web API dashboard.

You can access API analytics from the REST API Explorer. When exploring an API, click the context menu icon

then select **API analytics** to view analytics for that specific API.

For custom web services, such as scripted REST APIs or scripted SOAP web services, you can access analytics for the API by clicking the **API analytics** related link on the Scripted REST Service or Scripted Web Service forms.

When you directly access the analytics for an API, if there are no analytics for that specific API, the analytics for all APIs appears.

**REST & SOAP API analytics collection and cleanup**

API analytics uses scheduled jobs to collect and clean up analytics data.

The instance tracks all web service transactions for whitelisted APIs and maintains a daily history, aggregated by resource and HTTP action combination. Requester information is aggregated per requester, resource, and HTTP action combination and tracked up to the daily limit defined by the property `com.glide.api.stats.daily_limit`.

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
Refer to the following table to determine which requests are logged.

## Logged requests

<table>
<thead>
<tr>
<th>Whitelist API name</th>
<th>Example resource</th>
<th>Response code</th>
<th>Description</th>
<th>Logged</th>
</tr>
</thead>
<tbody>
<tr>
<td>now/table</td>
<td>/api/now/table/</td>
<td>Any except 401</td>
<td>Valid resource and table</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>incident</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>now/table</td>
<td>/api/now/table/</td>
<td>400</td>
<td>Valid resource but an invalid table</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>invalidResource</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>now/table</td>
<td>/api/now/table/</td>
<td>403</td>
<td>Requesting user has insufficient privileges</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>incident</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>now/table</td>
<td>/api/now/table/</td>
<td>401</td>
<td>Requesting user is not authenticated</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>incident</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>myApp/myScriptedApi</td>
<td>myApp/</td>
<td>Any except 401</td>
<td>Valid resource</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>myScriptedApi/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>myResource</td>
<td>myApp/invalidApiName</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>myApp/</td>
<td>400</td>
<td>Invalid API, even with a matching whitelist entry</td>
<td>No</td>
</tr>
<tr>
<td>myApp/invalidApiName</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

On the 2nd of each month, the API Monthly Stats scheduled job calculates the monthly total for each resource and HTTP action combination. Each day the API Monthly Requestor Stats scheduled job calculates the monthly total for each resource, requester, and HTTP action combination based on daily scores older than 2 days.

Daily statistics are maintained for 33 days. Monthly totals are maintained for 13 months. Table cleaners for the sys_api_stats, sys_api_stats_requestor, and sys_api_stats_requestor_monthly tables remove analytics records older than these limits.

### REST & SOAP API analytics naming

The **API Name** used when tracking API analytics is determined by the type of API being described, such as a REST API or a Scripted SOAP service.

<table>
<thead>
<tr>
<th>API type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST</td>
<td>The API namespace and the first part of the URI following the namespace is used as the API name.</td>
</tr>
<tr>
<td></td>
<td>For example, for the Table API endpoints api/now/table/incident and api/now/table/problem, the namespace and ID are now/table.</td>
</tr>
<tr>
<td>Direct SOAP (table does not extend Import Set Row table)</td>
<td>If the direct SOAP request accesses a table, Direct SOAP is used as the API name.</td>
</tr>
<tr>
<td>SOAP import (table extends Import Set Row table)</td>
<td>Import Set SOAP is used as the API name.</td>
</tr>
<tr>
<td>Scripted SOAP Services</td>
<td>The SOAP request endpoint page is used as the API name, such as my_service.do.</td>
</tr>
</tbody>
</table>
## API analytics properties

Certain properties control the behavior of API analytics.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| com.glide.api.stats.enabled | When true, enables the collection of API usage statistics. When false, no analytics are collected even for whitelisted domains.  
  - Type: true | false
  - Default value: true
  - Location: Add to the System Properties [sys_properties] table |
| com.glide.api.stats.max_bytes_nursery_size | The maximum amount of memory, in bytes, used to store transaction data before it is written to the log. Transaction data is written to the log regularly based on the value of `com.glide.api.stats.persist_interval`.  
  If a large volume of transactions exceeds this memory limit before the log is written, some transactions may not be logged. The event `api.stats.cache.size.reached` is fired if this limit is reached.  
  This property value must be between 1 and 3 megabytes.  
  - Type: integer
  - Default value: 3145728
  - Location: Add to the System Properties [sys_properties] table |
| com.glide.api.stats.persist_interval | The frequency, in seconds, for writing transactions stored in memory to the log. This property value must be between 30 and 120 seconds.  
  - Type: integer
  - Default value: 60
  - Location: Add to the System Properties [sys_properties] table |
| com.glide.api.stats.daily_limit | The daily limit of requestor stats records per instance node. As soon as the value is reached, data is no longer aggregated and stored for that day. The event `api.stats.requestor.daily.limit.reached` is fired if this limit is reached.  
  - Type: integer
  - Default value: 20000
  - Location: Add to the System Properties [sys_properties] table |
Outbound REST web service

ServiceNow outbound REST functionality allows you to retrieve, create, update, or delete data on a web services server that supports the REST architecture.

A REST message can be sent by a REST workflow activity or by using the RESTMessageV2 script API. You can run REST messages from a MID Server which allows the message to communicate with REST providers on an internal network.

ServiceNow REST functionality is flexible enough to accommodate many web service APIs. Be sure you are familiar with your web service and the parameters it accepts before attempting to define a REST message in ServiceNow.

The following video tutorial demonstrates how to configure outbound REST web service messages to consume third-party web services from the Now Platform.

REST message elements

An outbound REST message is composed of several elements, such as the endpoint and HTTP methods.

A REST message contains the following elements:

<table>
<thead>
<tr>
<th>Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint</td>
<td>The endpoint is the URL of the data to be retrieved, updated, or deleted. Every REST message must specify an endpoint.</td>
</tr>
<tr>
<td>Headers</td>
<td>HTTP headers in REST messages contain information about the request, such as the desired response format. A REST message may specify any number of headers.</td>
</tr>
<tr>
<td>Authentication settings</td>
<td>Authentication settings include which type of authentication to use, such as basic auth or OAuth, as well as the credentials to use.</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HTTP methods</td>
<td>HTTP methods, such as GET, POST, or DELETE interact with the data at the endpoint. You can optionally override the parent REST message configuration in each HTTP method such as by specifying a different endpoint, authentication credentials, or headers. HTTP methods that send content, such as POST, include a message body detailing this content. A REST message may specify multiple HTTP methods. When sending a REST message, such as through a workflow activity or script, you must specify which HTTP method to use.</td>
</tr>
</tbody>
</table>

Create a REST message

You can send requests to a REST web service endpoint by creating a REST message record.

Role required: web_service_admin

1. Navigate to **System Web Services > REST Message**.
2. Click **New**.
3. Complete the following fields:

   REST Message form

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a descriptive name for this message.</td>
</tr>
<tr>
<td>Endpoint</td>
<td>Enter the endpoint that this REST message is sent to. The endpoint value may include variables using the format <code>${variable}</code>.</td>
</tr>
<tr>
<td>Authentication type</td>
<td>Select the type of authentication to use, if any, and the profile record that contains the user credentials. Outbound REST supports basic authentication and OAuth 2.0. Outbound REST supports mutual authentication with basic authentication only. Authentication configured here is inherited by the associated HTTP methods. You can configure authentication for each method which overrides any authentication setting at the message level.</td>
</tr>
<tr>
<td>HTTP Headers</td>
<td>Double-click a row in the <strong>HTTP Headers</strong> embedded list to define the header Name and Value. The web service provider determines which headers are supported or required. See <strong>List of HTTP Header Fields</strong> for a list of HTTP header fields.</td>
</tr>
</tbody>
</table>
4. Click Submit.

After creating the REST message, a GET HTTP method is created automatically using the values from the REST message record.

Create or edit HTTP methods and run a request.

Define a REST message HTTP method

Define an HTTP method such as GET or POST to send a request to a web service provider.

Role required: web_service_admin

When you create a REST message record, several default HTTP methods are automatically created using settings inherited from the REST message record, such as the Endpoint. Subsequent changes to the REST message record are not applied to the HTTP methods automatically. You can create additional HTTP methods or modify the default HTTP methods to implement new behavior.

1. Navigate to System Web Services > Outbound > REST Message.
2. Select a REST message you want to define an HTTP method for.
3. In the HTTP Methods related list, click New.
4. Select the HTTP method you want to use, such as GET or POST.
5. Enter the Endpoint this HTTP method should access.
   - The endpoint value may include variables using the format ${variable}.
6. Right-click the form header and select Save.

After creating the HTTP method, you can override the security settings from the parent REST message, configure HTTP headers, add variables, or test the method. For PUT, POST, and PATCH methods you can define a message body.

Testing REST message HTTP methods

After configuring an HTTP method for an outbound REST message, you can test it to ensure that the request is valid and the response returns as expected.

To test an HTTP method, click the Test related link on the HTTP Method form.

Each test run displays the response status, such as 200 for a successful GET request, the full endpoint URL, any parameters passed in the request, and the response body.

Note: Fields on the Test Runs form are for information only; changes to these fields do not apply to the REST message or HTTP method. Do not modify these values when testing different REST message configurations. Instead, update the REST message or HTTP method, then run a new test.

If the HTTP method includes variables, the Test value for each variable in the Variable Substitutions related list is used when testing the method.

Completed test runs for an HTTP method appear in the Test Runs related list. If there was an error during the request, the Error Code and Error Message fields appear.

Define a REST message HTTP header

Define an HTTP header for a REST message or HTTP method to send that header with REST requests.

Role required: web_service_admin

You can specify an HTTP header for a REST message, or for an HTTP method. Headers defined for a REST message apply to all HTTP methods for that REST message. If you specify the same header for
both a REST message and a child HTTP method, the value defined for the HTTP method overrides the value from the parent REST message.

1. Navigate to System Web Services > REST Message.
2. Select a REST message.
3. Optional: To specify a header for an HTTP method instead of the REST message, in the HTTP Method related list, select an HTTP method.
4. Select the HTTP Request tab.
5. In the HTTP Headers embedded list, click Insert a new row.
6. Enter the name of the header, such as Content-Type or Accept.
   Supported headers depend on the REST web service provider you want to connect to. Refer to the documentation for your web service provider to identify which headers are valid or required.
7. Click on the Value field for the new row and enter the value you want to assign this header. You can use a variable in the format ${variable} instead of a static value. You can assign a value to the variable when sending a REST request.
8. Click Update.

Sending outbound REST messages through a MID Server
You can configure a REST message HTTP method to be sent through a MID Server.
By using a MID Server, the request can reach an endpoint that is behind a firewall or within a private network.
To configure an HTTP method to use a MID Server, select a MID Server in the Use MID Server field on the HTTP Method form. The instance must have an active MID Server to use this functionality.

Using special characters in URIs
A REST function URI or function variable may use special characters, such as pipe (|) characters.
When using these characters in a REST message, use URL encoding to escape these characters. For example, to use a parameter value of user|title, enter user%7Ctitle. Entering special characters directly may cause the REST message to fail and display the response Invalid uri <URI>: Invalid query.

Outbound REST authentication
Outbound REST messages support multiple types of authentication.
Different web service providers may require a specific type of authentication. Outbound REST supports the following authentication formats.
- Basic authentication using a username and password
- OAuth 2.0 using an OAuth provider and profile
- Mutual authentication using protocol profiles

Overriding REST authentication
You can define authentication for a REST message, or individually for each HTTP method. HTTP methods inherit authentication from their parent REST message record when the HTTP method Authentication type is Inherit from parent, which is the default value.
You can disable authentication for a specific HTTP method by setting the Authentication type field to **No authentication**, or specify authentication that is different from the parent REST message by selecting basic auth or OAuth.

### Authentication requirements

Outbound REST supports mutual authentication only when using basic authentication. Mutual authentication is not available with OAuth 2.0.

OAuth 2.0 can be used only with messages that are not configured to use a MID Server. You cannot send OAuth 2.0 authenticated messages through a MID Server.

When scripting new REST messages configured with authentication you must use the RESTMessageV2 API. The legacy RESTMessage APIs do not support current authentication formats.

### Configure a REST message with basic auth

You can configure an outbound REST message to provide basic authentication credentials with each request.

**Role required:** web_service_admin

Before starting this procedure, ensure there is a REST Message record that you want to configure to use basic auth.

#### Note:
Ensure any scripts that call this REST message use the RESTMessageV2 API. The RESTMessageV2 API is required to send authenticated REST messages via scripts.

1. Navigate to **System Web Services > Outbound > REST Message**.
2. Select a REST message record.
3. In the **Authentication type** field, select **Basic**.

#### Note: The Basic (Simple) choice appears on REST message records configured to use basic authentication prior to the Geneva release. This choice is intended for compatibility with older REST messages and should not be used for REST messages created in the Geneva release or later.

4. In the **Basic auth profile** field, select the basic authentication profile that contains the credentials you want to send.
5. Click **Submit**.

Test the REST message to ensure you receive the expected response. You can optionally specify different authentication settings for each HTTP method related to this REST message, overriding the parent REST message settings.

### Create a basic auth profile

Create a basic auth profile to specify basic authentication credentials for one or more REST messages.

**Role required:** web_service_admin

1. Navigate to **System Web Services > REST Message**.
2. Select a REST message record.
3. In the **Authentication type** field, select **Basic**.
4. In the **Basic auth profile** field, click the reference lookup icon.
5. Click **New**.

Enter a descriptive **Name** for the profile.
6. Enter the **Username** and **Password** you want to send as basic authentication credentials.

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
8. Click **Submit**.

Configure a REST message to use this basic auth profile.

**Configure a REST message with OAuth**

You can configure an outbound REST message to send OAuth credentials with the request.

Role required: web_service_admin and oauth_admin

Before starting this procedure, ensure:

- There is a REST Message record that you want to configure to use OAuth.
- There is an OAuth provider set up in the OAuth application registry with the OAuth client information to use.
- The OAuth provider has an associated OAuth 2.0 profile.
- The REST message HTTP Methods are not configured to use a MID Server.

**Note:** Ensure any scripts that call this REST message use the RESTMessageV2 API. The RESTMessageV2 API is required to send authenticated REST messages via scripts.

1. Navigate to **System Web Services > Outbound > REST Message**.
2. Select a REST message record.
3. In the **Authentication type** field, select **OAuth 2.0**.
4. In the **OAuth profile** field, select the OAuth 2.0 profile that specifies the credentials you want to send.
5. Right-click the form header and select **Save**.
   An info message appears at the top of the form indicating that you must request a new OAuth token.
6. Click the **Get OAuth Token** related link.
   Depending on your OAuth provider, a separate window from your provider may appear asking for confirmation before providing a token. Complete any steps required by the provider to obtain the token.

Test the REST message to ensure you receive the expected response. You can optionally specify different authentication settings for each HTTP method related to this REST message, overriding the parent REST message settings.

**Use a third-party OAuth provider**

Each client application must register with the instance to participate in OAuth authorization. You can use a third-party OAuth provider to authorize access.

Role required: admin

**Note:** ServiceNow only supports third-party OAuth providers to authorize requests from ServiceNow to third-party APIs.

For example, you might want to integrate with a third-party Calendar service which requires OAuth 2.0 access tokens to read a user’s scheduled events and create events. Configure the Calendar service as a third-party OAuth provider. This configuration allows you to get an access token from the Calendar service and then use the token to make requests against the service from ServiceNow.

1. Navigate to **System OAuth > Application Registry** and then click **New**.
2. On the interceptor page, click **Connect to a third party OAuth Provider** and then fill in the form.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A unique name that identifies the application to access.</td>
</tr>
<tr>
<td>Client ID</td>
<td>The unique ID of the application. The instance uses the client ID when requesting an access token. You must get the client ID from the authorization server.</td>
</tr>
<tr>
<td>Client Secret</td>
<td>(Required) The shared secret string that the instance and the application use to authorize communications with one another. If you do not enter the correct client secret, tokens are not issued.</td>
</tr>
<tr>
<td>OAuth API Script</td>
<td>This option enables you to reference an amended OAuthUtil script include. Copy and rename the default OAuthUtil script include file, and then amend this version for specific requests and responses to match your 3rd party OAuth provider. The amended script name must have the prefix OAuth. See OAuthUtil to add the required body parameter in the proper method.</td>
</tr>
<tr>
<td>Logo URL</td>
<td>The URL that contains an image to use as the application logo.</td>
</tr>
<tr>
<td>Default Grant Type</td>
<td>The type of grant:</td>
</tr>
<tr>
<td></td>
<td>• Authorization code: The code that is granted to the client to obtain an access token, which is then used to obtain access to the resource. If you select this option, then you need an authorization URL (the URL of the authorization server).</td>
</tr>
<tr>
<td></td>
<td>• Resource owner password credentials: The username and password of the user that is trying to obtain access to the resource.</td>
</tr>
<tr>
<td></td>
<td>• Client Credentials: The client ID and client secret, which are both used to get the access token. This method does not provide refresh tokens.</td>
</tr>
<tr>
<td>Refresh Token Lifespan</td>
<td>The refresh token lifespan in seconds.</td>
</tr>
<tr>
<td>Accessible from</td>
<td>The application scope that this registry is accessible from.</td>
</tr>
<tr>
<td>Active</td>
<td>A check box that indicates that the application registry is active.</td>
</tr>
<tr>
<td>Authorization URL</td>
<td>The URL of the endpoint to authorize the user if you are using the authorization code grant type. If you are accessing another ServiceNow instance, append /oauth_auth.do to the URL.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Token URL</td>
<td>The location of the token endpoint that the instance uses to retrieve and refresh tokens. If you are accessing another ServiceNow instance, append /oauth_token.do to the URL.</td>
</tr>
<tr>
<td>Redirect URL</td>
<td>The application endpoint that receives the authorization code. Leave the field empty to have the instance auto-generate the URL. If you are accessing another ServiceNow instance, append /oauth_redirect.do to the URL.</td>
</tr>
<tr>
<td>Token Revocation URL</td>
<td>The location of the endpoint that the instance uses to revoke the token. If you are accessing another instance, append /oauth_revoke.do to the URL.</td>
</tr>
<tr>
<td>Comments</td>
<td>Additional information to associate with the application.</td>
</tr>
<tr>
<td>Embedded lists</td>
<td></td>
</tr>
<tr>
<td>OAuth Entity Profiles</td>
<td>The profiles that are associated with the OAuth provider. The profile includes the grant type. Click the profile name to go to the OAuth Entity Profile form.</td>
</tr>
<tr>
<td>OAuth Entity Scopes</td>
<td>The entity scopes associated with the OAuth provider. The scope identifies the services the application has access to. Click the scope name to go to the OAuth Entity Scope form.</td>
</tr>
</tbody>
</table>

3. Click **Submit**. The record is saved in the Application Registries (oauth_entity) table.

The system creates a record in the Application Registries (oauth_entity) table of type **OAuth Provider**. The instance also auto-generates a default profile using the specified grant type, but without any scopes. You can create additional profiles, each with scopes.

**OAuth profiles and scopes**

In the OAuth provider scenario, profiles and scopes specify the grant type, authorization type, and level of access.

In the OAuth provider scenario, the OAuth profile refers to a combination of a grant type and at least one scope. The scope specifies the access that the user has to the protected resource, such as read or write. You can create a profile for each third-party provider and obtain the specific set of scopes from the provider. See Specify an OAuth profile and Specify an OAuth scope for more information. The instance also uses OAuth profiles when a REST call specifies OAuth 2.0 authentication. The instance auto-creates a default profile for each third-party provider record that you create. There can be only one default profile.

Specify the following parameters, which are saved in the OAuth Requestor Profile (OAuth_requestor_profile) table:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oauth_requestor</td>
<td>The sys_id of the object, which can be a user record or an email account.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>oauth_requestor_context</td>
<td>Descriptor that provides context for the OAuth requestor. As a good practice, use the name of the table where the oauth_requestor object is saved.</td>
</tr>
<tr>
<td>oauth_provider_profile</td>
<td>The sys_id of the default OAuth profile record (see Specify an OAuth profile).</td>
</tr>
</tbody>
</table>

When the user attempts to authenticate, the provider accesses the OAuth Requestor Profile table to look for the user. If the user is found, the authentication is successful. If not, the provider accesses the default profile to determine the grant type and how to proceed with the authentication.

Specify an OAuth profile
An OAuth profile includes the grant type that the third-party OAuth provider needs to obtain access to the restricted resource.

Role required: admin

1. Open a third-party OAuth provider record.
2. In the OAuth Entity Profiles embedded list, click Insert a new row and then enter a name for the profile.
3. Right-click the Application Registry form header and select Save. The system creates the profile record.
4. Click the name of the profile you created and then fill in the form fields.
**Field** | **Description**
---|---
Name | Enter a descriptive name.
OAuth provider | Verify the provider that is associated with the profile.
Grant type | Select the grant type:
- **Authorization code**: The code that is granted to the client to obtain an access token, which is then used to obtain access to the resource. If you select this option, then you need an authorization URL (the URL of the authorization server).
- **Resource owner password credentials**: The username and password of the user that is trying to obtain access to the resource.
- **Client Credentials**: The client ID and client secret, which are both used to get the access token. This method does not provide refresh tokens.
5. Click Update.

Specify an OAuth scope

Specify the OAuth scopes that you get from the provider. Scopes can be any level of access specified by the provider, such as read, write, or any string, including a URL.

Role required: admin

1. Open a third-party OAuth provider record.
2. Open a profile associated with the provider.
3. In the OAuth Entity Profile Scopes embedded list, click Insert a new row and then enter a Name for the profile.
4. Right-click the OAuth Entity Profile form header and select Save. The profile record is created.
5. Click the name of the scope that you created and then fill in the form fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a descriptive name.</td>
</tr>
<tr>
<td>OAuth provider</td>
<td>Verify the provider associated with this scope.</td>
</tr>
<tr>
<td>OAuth scope</td>
<td>The scope that you are granted by the provider. Typical scopes are read and write. Scopes can be any string that the provider specifies.</td>
</tr>
</tbody>
</table>

6. Click Update.
Outbound REST with OAuth 2.0 profile tutorial - integrating with Google Contacts API

This tutorial explains how to use an OAuth 2.0 profile to authenticate an outbound REST message with Google to retrieve contact information.

The procedures detailed in this tutorial require the oauth_admin and web_service_admin roles. Ensure you have both of these roles before starting this tutorial.

OAuth 2.0 tutorial - configure the Google service as an OAuth provider

Use the Google Developer Console to set up an OAuth 2.0 provider.

Role required: None

This procedure is performed within the Google Developer Console. You must have a Google account to access this console.

Configure the Google service in order to obtain a client ID and client secret, and specify your ServiceNow instance URL as the OAuth redirect URL.

Note: This information describes the state of the Google Developer Console and Contacts API as of July 22, 2015. Changes made after that date may not be included in this document.

2. Log in using your Google credentials.
3. Click Select a project.
4. Click Create a project.
5. Enter a Project name.
6. Click Create.
   After Google creates the project, the project dashboard appears.
7. Navigate to APIs & auth > APIs.
8. Select the Contacts API.
9. Click Enable API.
10. Navigate to APIs & auth > Credentials.
11. Click Create new Client ID.
12. Ensure the Web application radio button is selected and click Configure consent screen.
13. Enter a descriptive Product name.
   This name appears when you authorize the OAuth token in your instance.
14. Click Save.
15. In the Create Client ID window, add the OAuth redirect URI for your instance to the Authorized redirect URIs field.
   This URL follows the format https://<instance>.service-now.com/oauth_redirect.do
16. Click Create Client ID.
   The client ID information appears.
17. Record the Client ID and Client secret values.
   You will need these values to configure the Google service as an OAuth provider in your instance.

OAuth 2.0 tutorial - create an OAuth provider and profile

Set up the Google service as an OAuth provider in ServiceNow by entering your client information, Google API URLs, and configuring the OAuth profile.

Role required: oauth_admin
You must have configured the Google service as an OAuth provider and recorded your **Client ID** and **Client Secret** values.

1. Navigate to **System OAuth** > **Application Registry**.
2. Click **New**.
3. Select **Connect to a third party OAuth Provider**.
4. Enter a **Name** for the OAuth provider. For this example, use Google.
5. Enter the **Client ID** and **Client Secret** that you obtained from Google.
6. Set the **Default Grant type** to **Authorization Code**.
9. In the **Redirect URL** field, enter `https://<instance>.service-now.com/oauth_redirect.do`
   This URL must match the redirect URL provided to Google.
10. In the **Token Revocation URL** field, enter `https://accounts.google.com/o/oauth2/revoke`.
11. Right-click the form header and select **Save**.
   A new OAuth Entity Profile record is created.
12. In the **OAuth Entity Scopes** embedded list, add a new row with the **Name** and **OAuth scope** values set to `https://www.googleapis.com/auth/contacts.readonly`.
13. Right-click the form header and select **Save**.
14. In the **OAuth Entity Profiles** embedded list, select the automatically-created profile.
15. In the **OAuth Entity Profile Scopes** embedded list, add a new row and select the Google contacts API read-only scope.
16. Click **Update**.

**OAuth 2.0 tutorial - create a REST message**

Create a REST message and associated HTTP method to contact the Google service using the OAuth 2.0 profile.

Role required: web_service_admin and oauth_admin

You must have configured an OAuth provider and profile using the Google API information and your OAuth credentials.

1. Navigate to **System Web Services** > **REST Message**.
2. Click **New**.
3. Enter a descriptive **Name**.
4. In the **Endpoint** field, enter `https://www.google.com/m8/feeds/contacts/default/full`.
   By using `default` instead of a specific username, the Google API uses the OAuth credentials to determine which account to get information from.
5. In the **Authentication** tab, set the **Authentication type** to **OAuth 2.0**.
6. In the **OAuth profile** field, select the Google contacts OAuth profile.
7. Right-click the form header and select **Save**.
8. Click the **Get OAuth Token** related link to request an authorization token from Google using the configured client ID and secret.
9. In the Request for Permission window that appears, click **Accept** to grant access to your Google contacts.
   The token acquired is not directly accessible in your instance.
10. In the **HTTP Methods** related list, select the **GET** method.
11. Leave the HTTP method **Authentication type** as **-- None --** to use the OAuth profile from the parent REST message record.
12. On the **HTTP Request** tab, add a new row to the **HTTP Headers** related list with a **Name** of **GData - Version** and a **Value** of **3.0**.
13. Right-click the form header and select **Save**.

14. Click the **Test** related link.

   The test result should display an **HTTP Status** of 200, and the result of the contacts API call.

---

**Outbound REST mutual authentication**

Mutual authentication causes the web service provider and consumer to authenticate with each other before communicating.

ServiceNow Technical Supports mutual authentication for outbound web services. Mutual authentication is not available for inbound web services or for outbound web services that use a MID Server.

---

**Variable substitution in outbound REST messages**

You can use variables when creating outbound REST messages and assign values to those variables when performing a request.

Variables are allowed in the **Endpoint URL**, **HTTP Header** and **HTTP Query Parameter Value** fields, and the **Content** field for POST and PUT methods.

The syntax for variables is `${variable_name}`. The REST message substitutes this variable with the parameter values provided when the method runs. For example, if the REST message **Endpoint** is `http://myserver.mycompany.com/offices/${id}`, a parameter named **id** must exist and contain a value that can be used when the method runs.

You can assign a value to variables when running the request using the RESTMessageV2 API `setStringParameter` and `setStringParameterNoEscape` methods.

When testing an HTTP method that includes variables, the **Test value** for each variable in the **Variable Substitutions** related list is used.

---

**Generate REST message variables**

You can automatically populate the list of variable substitutions, based on variables defined in several REST message HTTP method fields.

Role required: web_service_admin

Before starting this procedure, create a REST Message record with at least one HTTP method that uses variables.

1. Navigate to **System Web Services > REST Message**.
2. Select a REST Message record.
3. Select a method from the **HTTP Methods** related list.
4. Click the **Auto-generate variables** related link.

   The **Variable Substitutions** related list is automatically populated for any variables defined in the HTTP Method **Endpoint** field and the **HTTP Headers** and **HTTP Query Parameters** embedded lists. For POST and PUT messages, variables defined in the **Content** field are also used.

You can use the REST Message workflow activity to send the message, or click **Preview Script Usage** to get a sample script. The sample script includes a `setStringParameter` call for each defined variable substitution that allows you to assign a value to the variable in your script.

---

**Scripting outbound REST**

You can send outbound REST requests from any place in the Now Platform where scripting is allowed.
For example, you can return data from a REST endpoint using a business rule when an event is triggered. Create a script from scratch or let the REST message preview feature create the script based on content and parameters you provide in the method record.

For detailed API information about the server-side RESTMessageV2 and RESTResponseV2 APIs, see the API documentation on developer.servicenow.com.

Generate a REST message script preview

You can generate an example script to send a REST message based on content and parameters you provide in the method record.

Role required: web_service_admin or admin

Generate an example script and use it as a starting point when scripting outbound REST messages.

1. Navigate to System Web Services > REST Message.
2. Select a REST message record.
3. In the HTTP Methods related list, select an HTTP method record.
4. Ensure the HTTP method is configured as needed, including any variables.
5. Save the record.
6. In the Variable Substitutions related list, assign a value to each variable.
7. Under Related Links, click Preview script usage.

The instance displays the script that the REST message generated for this method.

8. Copy this script and modify it as needed to use elsewhere in the instance.
Refer to the RESTMessageV2 and RESTResponseV2 APIs for more information on available scripting methods. Outbound REST scripting examples are also available.

**Direct RESTMessageV2 example**

You can send an outbound REST message directly to the endpoint.

In this example, the script sends a REST message requesting a stock quote and waits for a response. If there is no response from the web service provider, or if the specified REST message record is unavailable, the script throws an error, handled in this example by the try-catch block.

```javascript
var requestBody;
var responseBody;
var status;
var sm;
try{
  sm = new sn_ws.RESTMessageV2("Yahoo Finance", "get"); // Might throw exception if message doesn't exist or not visible due to scope.
  sm.setBasicAuth("admin","admin");
  sm.setStringParameter("symbol", "NOW");
  sm.setStringParameterNoEscape("xml_data", "<data>test</data>" );
  sm.setHttpTimeout(10000); //In milliseconds. Wait at most 10 seconds for response from http request.
  response = sm.execute(); //Might throw exception if http connection timed out or some issue with sending request itself because of encryption/decryption of password.
  responseBody = response.haveError() ? response.getErrorMessage() : response.getBody();
  status = response.getStatusCode();
} catch(ex) {
  responseBody = ex.getMessage();
  status = '500';
} finally {
  requestBody = sm ? sm.getRequestBody():null;
}
gs.info("Request Body: " + requestBody);
gs.info("Response: " + responseBody);
gs.info("HTTP Status: " + status);
```

**Asynchronous RESTMessageV2 example**

You can send an outbound REST message asynchronously.

You can send a REST message asynchronously. When you send an asynchronous message the instance does not wait for a response before proceeding. You must handle waiting for a response within your code.

```javascript
var requestBody;
var responseBody;
var status;
var sm;
try{
  sm = new sn_ws.RESTMessageV2("Yahoo Finance", "get"); // Might throw exception if message doesn't exist or not visible due to scope.
  sm.setBasicAuth("admin","admin");
  sm.setStringParameter("symbol", "NOW");
  sm.setStringParameterNoEscape("xml_data", "<data>test</data>" );
  response = sm.executeAsync(); //Might throw exception if http connection timed out or some issue with sending request itself because of encryption/decryption of password.
```
response.waitForResponse(60); // In seconds. Wait at most 60
seconds to get response from ECC Queue/Mid Server //Might throw
exception timing out waiting for response in ECC queue.

responseBody = response.haveError() ?
response.getErrorMessage() : response.getBody();
status = response.getStatusCode();
} catch(ex) {
responseBody = ex.getMessage();
status = '500';
} finally {
requestBody = sm ? sm.getRequestBody():null;
}
gs.log("Request Body: " + requestBody);
gs.log("Response: " + responseBody);
gs.log("HTTP Status: " + status);

**RESTMessageV2 MID server example**

You can send an outbound REST message through a MID Server.

By sending the message through a MID Server, you can access endpoints that are behind a
firewall or within a private network. All REST messages sent through a MID Server are asynchronous.

var requestBody;
var responseBody;
var status;
var sm;
try{
sm = new sn_ws.RESTMessageV2("Yahoo Finance", "get");  // Might
throw exception if message doesn't exist or not visible due to
scope.
sm.setBasicAuth("admin","admin");
sm.setStringParameter("symbol", "NOW");
sm.setStringParameterNoEscape("xml_data","<data>test</data>");
sm.setMIDServer('mid_server_name');
response = sm.executeAsync(); // Might throw exception if http
connection timed out or some issue with sending request itself
because of encryption/decryption of password.

response.waitForResponse(60); // In seconds. Wait at most 60
seconds to get response from ECC Queue/Mid Server //Might throw
exception timing out waiting for response in ECC queue.

responseBody = response.haveError() ?
response.getErrorMessage() : response.getBody();
status = response.getStatusCode();
} catch(ex) {
responseBody = ex.getMessage();
status = '500';
} finally {
requestBody = sm ? sm.getRequestBody():null;
}
gs.info("Request Body: " + requestBody);
gs.info("Response: " + responseBody);
gs.info("HTTP Status: " + status);

**Note:** This example uses `waitForResponse` to pause for a response,
and then details how the response is handled. However, when using
Recordless RESTMessageV2 example

You can use the RESTMessageV2() constructor with no parameters to define a REST message entirely in the script.

When using this constructor you must provide an endpoint and HTTP method. In this example, the script creates an empty REST message and sets the values needed to insert an incident record.

```javascript
var restMessage = new sn_ws.RESTMessageV2();
restMessage.setBasicAuth("admin", "admin");
restMessage.setHttpMethod("post");
restMessage.setEndpoint("http://<instance>.service-now.com/api/now/table/incident");
restMessage.setRequestBody("{"short_description" : "Test incident"}");
var response = restMessage.execute();
```

Outbound SOAP web service

The SOAP Message module can be used to develop, prototype, and save outbound SOAP messages that can be reused in business rules and scripts.

You can use outbound SOAP messages in scripts using the SOAPMessageV2 API and the SOAPResponseV2 API. Examples detailing how to script outbound SOAP are available.

Outbound SOAP video tutorial

The following video tutorial demonstrates how to configure outbound SOAP web service messages to consume third-party web services from an instance.

SOAP message

Information needed to send SOAP requests is stored in SOAP message records.

Each record specifies an endpoint for the request, the required format of the request as a web services description language (WSDL) file, authentication information, and a list of functions that can run against the endpoint.

Create a SOAP message

Create a SOAP message to define the remote endpoint, WSDL, and authentication settings.

Role required: web_service_admin

1. Navigate to System Web Services > SOAP Message.
2. Click New.
3. Enter a Name to identify the SOAP message.
4. Specify a WSDL using one of these options:
   - To download and use an online WSDL source, select the Download WSDL check box and enter the URL for the WSDL in the WSDL field.
   - To enter the WSDL directly, clear the Download WSDL check box, and then copy and paste the WSDL XML into the WSDL XML field.

executeAsync, consider processing the response body in a separate business rule to take advantage of the asynchronous call rather than using waitForResponse.
5. If the endpoint is protected by basic authentication, select the **Use basic auth** check box and enter the credentials.

6. If the endpoint requires mutual authentication, select the **Enable mutual authentication** check box and select a Protocol profile to use for mutual authentication.

7. Click **Submit**.

This image shows an example of a SOAP message that connects to a demo instance of ServiceNow.
### SOAP Message

<table>
<thead>
<tr>
<th>Required field</th>
<th>Update</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: demo1 incident</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSDL: <a href="https://demo1.service-now.com/incident.do?WSDL">https://demo1.service-now.com/incident.do?WSDL</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description: Sample ServiceNow direct web service</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSDL XML</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```xml
    <soap:Body>
        <GetKeys>
            <key xsi:type="xsd:string">key1</key>
            <key xsi:type="xsd:string">key2</key>
        </GetKeys>
    </soap:Body>
</soap:Envelope>
```

### Related Links

- [Generate sample SOAP messages](#)

### SOAP Message Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>SOAP action</th>
<th>Envelope</th>
<th>Lock</th>
<th>Updated</th>
</tr>
</thead>
</table>

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
SOAP message functions

After you create a SOAP message record, you can click **Generate sample SOAP messages** to populate the **SOAP Message Functions** related list.

The instance creates these functions by reading the supplied WSDL definition.
**SOAP Message Function**

**Function:** insert

**SOAP message:** demo1 incident

**Basic auth user ID:** itil

**Basic auth user password:** ********

**Lock:**

**Use basic auth:**

**SOAP action:** http://www.service-now.com/incident/insert

**SOAP endpoint:** https://demo1.service-now.com/incident.do?SOAP

---

**Related Links**

- Preview script usage
- Test

---

**SOAP Message Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>short_description</td>
<td>this is the short description</td>
</tr>
</tbody>
</table>

---

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
The **SOAP action**, **SOAP endpoint**, and **Envelope** fields should be populated automatically based on the WSDL definition. The **Envelope** defines the message to send to the endpoint. In this example, the **Envelope** values have this format:

```
...<!-- optional --><short_description xsi:type="xsd:string">String</short_description>
...
```

To submit a specific value, enter the value directly in the appropriate XML tag. In this example, to set the **Short description** for a record, enter:

```
...
<short_description xsi:type="xsd:string">This is the short description</short_description>
...
```

### Variable substitution in outbound SOAP

To use variable substitution, use the format `${<variable_name>}` instead of defining a specific value.

```
...
<short_description xsi:type="xsd:string">${short_desc}</short_description>
...
```

To test variable substitution after you have modified the SOAP envelope with the variables, define values for the variables in the **SOAP Message Parameters** related list. For example, click **New** and enter the following information:

![Soap Message Parameters](image)

**Soap message parameters**

### Test the SOAP message

Test a SOAP message to validate the configuration before using the message in an integration. To test the SOAP message, click the **Test** related link. You are redirected to a test result form as shown below.
Soap message test

You can see the original SOAP request message, the resulting HTTP status code, and the SOAP response in this screen. You can also click the Rerun test related link to resubmit the SOAP request.

Note: A test SOAP message will time out after 60 seconds if a response is not received.

Send a SOAP message through a MID server

When creating SOAP message functions, you can configure the function to be sent through a MID Server.

There must be a running MID Server associated with your instance to use this functionality. All SOAP messages sent through a MID Server are performed asynchronously.
Soap message mid

By specifying a MID Server, all SOAP requests that use this SOAP message are sent through that MID Server. You can override the selected MID Server by using the `setMIDServer(mid server)` API call in a script.

Create a SOAP message from a WSDL that references an external XSD file

Create a SOAP message from a WSDL and external XSD file.

Role required: web_service_admin

This task includes example WSDL and XSD files for a weather forecast SOAP message. Your WSDL and XSD file will vary.

1. Navigate to System Web Services > SOAP Message and create a new record.
2. Clear the Download WSDL check box.
3. Paste the content of the WSDL into the WSDL XML field.
4. Save the record.
5. In the SOAP Message Imports related list, click New.
6. Paste the content of the XSD file into the **External Document** field.
7. Set the **Schema Location** field to `db://<name of the referenced XSD file>.xsd`.
   Specifying the schema location allows the instance to know the location of the referenced XSD file.
8. Click **Submit**.
9. Click **Generate sample SOAP messages**.

**Example WSDL and XSD files**

```xml
<?xml version="1.0" encoding="utf-8"?>
xmlns:xsd=":http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsd1/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsd1/http/"
targetNamespace="http://www.webservicex.net"
xmlns:wssdl="http://schemas.xmlsoap.org/wsd1/">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsd1.documentation">Get one week weather forecast for valid zip code or Place name in USA</wsdl:documentation>
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://www.webservicex.net">
      <s:include schemaLocation="WeatherForecast.xsd" />
    </s:schema>
  </wsdl:types>
  <wsdl:message name="GetWeatherByZipCodeSoapIn">
    <wsdl:part name="parameters" element="tns:GetWeatherByZipCode" />
  </wsdl:message>
  <wsdl:message name="GetWeatherByZipCodeSoapOut">
    <wsdl:part name="parameters" element="tns:GetWeatherByZipCodeResponse" />
  </wsdl:message>
  <wsdl:message name="GetWeatherByPlaceNameSoapIn">
    <wsdl:part name="parameters" element="tns:GetWeatherByPlaceName" />
  </wsdl:message>
  <wsdl:message name="GetWeatherByPlaceNameSoapOut">
    <wsdl:part name="parameters" element="tns:GetWeatherByPlaceNameResponse" />
  </wsdl:message>
  <wsdl:message name="GetWeatherByZipCodeHttpGetIn">
    <wsdl:part name="ZipCode" type="s:string" />
  </wsdl:message>
  <wsdl:message name="GetWeatherByZipCodeHttpGetOut">
    <wsdl:part name="Body" element="tns:WeatherForecasts" />
  </wsdl:message>
  <wsdl:message name="GetWeatherByPlaceNameHttpGetIn">
    <wsdl:part name="PlaceName" type="s:string" />
  </wsdl:message>
  <wsdl:message name="GetWeatherByPlaceNameHttpGetOut">
    <wsdl:part name="Body" element="tns:WeatherForecasts" />
  </wsdl:message>
  <wsdl:message name="GetWeatherByZipCodeHttpPostIn">
    <wsdl:part name="ZipCode" type="s:string" />
  </wsdl:message>
  <wsdl:message name="GetWeatherByZipCodeHttpPostOut">
    <wsdl:part name="Body" element="tns:WeatherForecasts" />
  </wsdl:message>
</wsdl:definitions>
```
<wsdl:message name="GetWeatherByPlaceNameHttpPostIn">
  <wsdl:part name="PlaceName" type="s:string" />
</wsdl:message>

<wsdl:message name="GetWeatherByPlaceNameHttpPostOut">
  <wsdl:part name="Body" element="tns:WeatherForecasts" />
</wsdl:message>

<wsdl:portType name="WeatherForecastSoap">
  <wsdl:operation name="GetWeatherByZipCode">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get one week weather forecast for a valid Zip Code (USA)"</wsdl:documentation>
    <wsdl:input message="tns:GetWeatherByZipCodeSoapIn" />
    <wsdl:output message="tns:GetWeatherByZipCodeSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="GetWeatherByPlaceName">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get one week weather forecast for a place name (USA)"</wsdl:documentation>
    <wsdl:input message="tns:GetWeatherByPlaceNameSoapIn" />
    <wsdl:output message="tns:GetWeatherByPlaceNameSoapOut" />
  </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="WeatherForecastHttpGet">
  <wsdl:operation name="GetWeatherByZipCode">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get one week weather forecast for a valid Zip Code (USA)"</wsdl:documentation>
    <wsdl:input message="tns:GetWeatherByZipCodeHttpGetIn" />
    <wsdl:output message="tns:GetWeatherByZipCodeHttpGetOut" />
  </wsdl:operation>
  <wsdl:operation name="GetWeatherByPlaceName">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get one week weather forecast for a place name (USA)"</wsdl:documentation>
    <wsdl:input message="tns:GetWeatherByPlaceNameHttpGetIn" />
    <wsdl:output message="tns:GetWeatherByPlaceNameHttpGetOut" />
  </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="WeatherForecastHttpPost">
  <wsdl:operation name="GetWeatherByZipCode">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get one week weather forecast for a valid Zip Code (USA)"</wsdl:documentation>
    <wsdl:input message="tns:GetWeatherByZipCodeHttpPostIn" />
    <wsdl:output message="tns:GetWeatherByZipCodeHttpPostOut" />
  </wsdl:operation>
  <wsdl:operation name="GetWeatherByPlaceName">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get one week weather forecast for a place name (USA)"</wsdl:documentation>
    <wsdl:input message="tns:GetWeatherByPlaceNameHttpPostIn" />
    <wsdl:output message="tns:GetWeatherByPlaceNameHttpPostOut" />
  </wsdl:operation>
</wsdl:portType>
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="GetWeatherByZipCode">
  <wsdl:input>
    <soap:body use="literal" />
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal" />
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetWeatherByPlaceName">
  <soap:operation soapAction="http://www.webservicex.net/GetWeatherByPlaceName" style="document" />
  <wsdl:input>
    <soap:body use="literal" />
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal" />
  </wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:binding name="WeatherForecastSoap12" type="tns:WeatherForecastSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="GetWeatherByZipCode">
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetWeatherByPlaceName">
    <soap12:operation soapAction="http://www.webservicex.net/GetWeatherByPlaceName" style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="WeatherForecastHttpGet" type="tns:WeatherForecastHttpGet">
  <http:binding verb="GET" />
  <wsdl:operation name="GetWeatherByZipCode">
    <http:operation location="/GetWeatherByZipCode" />
    <wsdl:input>
      <http:urlEncoded />
    </wsdl:input>
    <wsdl:output>
      <mime:mimeXml part="Body" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetWeatherByPlaceName">
    <http:operation location="/GetWeatherByPlaceName" />
    <wsdl:input>
      <http:urlEncoded />
    </wsdl:input>
    <wsdl:output>
      <mime:mimeXml part="Body" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="ZipCode" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
<s:complexType>
  <s:element name="GetWeatherByZipCodeResponse">
    <s:complexType>
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="GetWeatherByZipCodeResult" type="tns:WeatherForecasts" />
      </s:sequence>
    </s:complexType>
  </s:element>
</s:complexType>
<s:complexType name="WeatherForecasts">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="Latitude" type="s:float" />
    <s:element minOccurs="1" maxOccurs="1" name="Longitude" type="s:float" />
    <s:element minOccurs="1" maxOccurs="1" name="AllocationFactor" type="s:float" />
    <s:element minOccurs="0" maxOccurs="1" name="FipsCode" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="PlaceName" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="StateCode" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Status" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Details" type="tns:ArrayOfWeatherData" />
  </s:sequence>
</s:complexType>
<s:complexType name="ArrayOfWeatherData">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded" name="WeatherData" type="tns:WeatherData" />
  </s:sequence>
</s:complexType>
<s:complexType name="WeatherData">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Day" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="WeatherImage" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="MaxTemperatureF" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="MinTemperatureF" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="MaxTemperatureC" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="MinTemperatureC" type="s:string" />
  </s:sequence>
</s:complexType>
<s:element name="GetWeatherByPlaceName">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="PlaceName" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
Demonstration video

This video shows examples of creating and sending SOAP messages. The script example in this video uses the version one SOAPMessage API.

Connectivity details

For the ServiceNow-initiated SOAP requests to successfully communicate with the web service provider inside a remote network, the ServiceNow instance must have HTTP or HTTPS access to the SOAP endpoint at the provider.

Like any integration, such as LDAP, web services, or JDBC, the SOAP endpoint may reside behind a firewall that is blocking inbound communication from the instance. If this is the case, you need to make network changes to allow this connectivity into your network. You can either modify the firewall and ACL rules to allow the instance IP address, configure the SOAP message to use a MID Server, or implement a VPN tunnel to allow the instance communication into your network.

Note: A common misconception is that because asynchronous SOAP requests are routed through the ECC queue, they are always sent through a MID Server. This is not the case. Asynchronous SOAP requests only use a MID Server when configured to do so.

Outbound SOAP security

You can authenticate outbound SOAP messages using several different security protocols. The security protocol you should use depends on the requirements of the web service provider. Mutual authentication is supported for outbound web services.

Enable basic authentication for outbound SOAP

If the endpoint requires a user name and password, you can provide credentials using basic authentication.

1. Navigate to System Web Services > Outbound > SOAP Message.
2. Select a SOAP message record.
3. In the SOAP Message Functions related list, select a function.
4. Select Use basic auth.
5. Enter a user name in the Basic auth user ID field.
6. Enter the password for that user in Basic auth user password.
7. Click Update.
Enable WS-Security for outbound SOAP

You can sign outbound SOAP messages using username and password or a key store and trusted server certificate saved on the instance.

Role required: admin

1. Navigate to System Web Services > Outbound > SOAP Message.
2. Select a SOAP message record.
3. In the SOAP Message Functions related list, select a function.
4. In the WS-Security type field select the type of credentials to use, such as Username or X.509.
   Some web service providers may require you to submit both types of credential.
5. In the WS-Security x.509 profile or WS-Security Username profile field, select the profile that contains the credentials you want to use.
   If the record was configured with x.509 security prior to the Helsinki release, you can migrate the security settings to a WS-Security x.509 profile record by clicking the Migrate to X509 Security Profile button. When you click this button, security configuration field values (Key store, Certificate, Key store alias, and Password) are migrated to a new WS-Security x.509 profile record and that record is referenced from the SOAP message function. If an x.509 profile record already exists with matching values, that record is referenced.

   Note: Existing SOAP message functions that use the legacy security configuration will still work, however new records can use only an x.509 profile record for x.509 security.

6. Click Update.
   Credentials from the selected profile are sent as part of the SOAP message header.

Configure SOAP with a proxy

Certain properties provide support for SOAP requests to use a web proxy server.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>glide.http.proxy_host</td>
<td>The proxy server hostname or IP address</td>
<td>proxy.company.com, 192.168.34.54</td>
</tr>
<tr>
<td>glide.http.proxy_port</td>
<td>The port number for the proxy server</td>
<td>8080, 9100</td>
</tr>
<tr>
<td>glide.http.proxy_username</td>
<td>If the proxy server is authenticating using user name and password, enter a value for this property</td>
<td>proxyuser</td>
</tr>
<tr>
<td>glide.http.proxy_password</td>
<td>If the proxy server is authenticating using user name and password, enter a value for this property</td>
<td>password</td>
</tr>
</tbody>
</table>

Scripting outbound SOAP

You can send outbound SOAP requests from any place in the Now Platform where scripting is allowed.
For detailed API information about the server-side SOAPMessageV2 and SOAPResponseV2 APIs, see the API documentation on developer.servicenow.com.

**Preview a SOAP message script**

Generate a sample script to send the SOAP message.

After you have developed and tested the SOAP message, click the Preview script usage related link in the SOAP Message Function form. The dialog box displays an example of how you can invoke the SOAP message from a script.

You can manipulate the resulting XML response body with `XMLDocument` or with `gs.getXMLText` and `gs/XMLNodeList`.

**Direct SOAPMessageV2 example**

You can send an outbound SOAP message directly to the endpoint.

In this example, the script sends a SOAP message requesting a stock quote and waits for a response. If there is no response from the web service provider, or if the specified SOAP message record is unavailable, the script throws an error, handled in this example by the try-catch block.

```javascript
var requestBody;
var responseBody;
var status;
var sm;
try{
    sm = new sn_ws.SOAPMessageV2("StockQuote", "GetQuote"); // Might throw exception if message doesn't exist or not visible due to scope
    sm.setBasicAuth("admin","admin");
    sm.setStringParameter("symbol", "NOW");
    sm.setStringParameterNoEscape("xml_data","<data>test</data>");
    sm.setHttpTimeout(10000) //In Milli seconds. Wait at most 10 seconds for response from http request.
    response = sm.execute(); //Might throw exception if http connection timed out or some issue with sending request itself because of encryption/ decryption of password and stuff
    responseBody = response.haveError() ? response.getErrorMessage() :
        response.getStatusCode();
    status = response.getStatusCode();
} catch(ex) {
    responseBody = ex.getMessage();
    status = '500';
} finally {
    requestBody = sm ? sm.getRequestBody():null;
}
    gs.info("Request Body: " + requestBody);
    gs.info("Response: " + responseBody);
    gs.info("HTTP Status: " + status);
```

**Asynchronous SOAPMessageV2 example**

You can send an outbound SOAP message asynchronously.

When you send an asynchronous message the instance does not wait for a response before proceeding. You must handle waiting for a response within your code.

```javascript
var requestBody;
var responseBody;
var status;
var sm;
```
try{
    sm = new sn_ws.SOAPMessageV2("StockQuote", "GetQuote");  // Might throw exception if message doesn't exist or not visible due to scope
    sm.setBasicAuth("admin","admin");
    sm.setStringParameter("symbol", "NOW");
    sm.setStringParameterNoEscape("xml_data","<data>test</data>");
    response = sm.executeAsync(); //Might throw exception if http connection timed out or some issue with sending request itself because of encryption/decryption of password

    response.waitforResponse(60); // In Seconds, Wait at most 60 seconds to get response from ECC Queue/Mid Server //Might throw exception timing out waiting for response in ECC queue

    responseBody = response.haveError() ? response.getErrorMessage() : response.getBody();
    status = response.getStatusCode();
} catch(ex) {
    responseBody = ex.getMessage();
    status = '500';
} finally {
    requestBody = sm ? sm.getRequestBody():null;
}

gs.info("Request Body: " + requestBody);
gs.info("Response: " + responseBody);
gs.info("HTTP Status: " + status);

SoapMessageV2 MID server example

You can send an outbound SOAP message through a MID Server.

By sending the message through a MID Server, you can access endpoints that are behind a firewall or within a private network. All SOAP messages sent through a MID Server are asynchronous.
Recordless SOAPMessageV2 example

You can use the SOAPMessageV2() constructor with no parameters to define a SOAP message entirely in the script.

When using this constructor you must provide an endpoint and SOAP action. In this example, the script creates an empty SOAP message and sets the values needed to insert an incident record.

```javascript
var s = new sn_ws.SOAPMessageV2(); //create an empty SOAP message
s.setBasicAuth('admin','admin');
s.setSOAPAction('http://www.service-now.com/incident/insert'); //set the SOAP action to perform
s.setEndpoint('http://<instance>.service-now.com/incident.do?SOAP'); //set the web service provider endpoint
var response = s.execute();
var xmldoc = new XMLDocument(response.getBody());
var incident_sysid = xmldoc.getNodeText('//sys_id');
```

Outbound web services mutual authentication

Mutual authentication establishes trust by exchanging Secure Sockets Layer (SSL) certificates. Before connecting to a server, the client requests an SSL certificate. The server responds by requesting that the client send its own certificate. Both respond by validating the certificates of the other and sending acknowledgments before initiating an HTTPS connection. Mutual authentication is not available for inbound requests or for outbound web service calls through a MID Server.

As an administrator, you can enable mutual authentication by defining a protocol profile for connections that require mutual authentication. Protocol profiles allow you to associate a specific certificate record with a protocol, such as HTTPS.

For example, you can create a protocol profile called `https` for one-way SSL and another called `myhttps` with a certificate for mutual authentication. You can then make an HTTPS web service request by calling `myhttps://<externalendpoint.com>` if the end point requires mutual authentication, or `https://<externalendpoint.com>` if it does not.

### Note:
This feature enables mutual authentication only on outbound HTTPS connections, such as SOAP, REST, or direct HTTPS calls. ServiceNow does not support mutual authentication for inbound requests or for outbound requests sent through a MID Server.

Create a protocol profile

You can create a protocol profile for outbound web services, such as to enable mutual authentication.

Role required: admin

2. Click New.
3. Fill in the fields on the form, as appropriate.

### Name of form

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>Enter a unique name to identify this protocol, such as <code>myhttps</code>. The protocol name allows you to differentiate between normal HTTPS connections and HTTPS connections that use this protocol profile.</td>
</tr>
<tr>
<td>Default port</td>
<td>Enter the port number for connections that use this protocol.</td>
</tr>
<tr>
<td>Check box</td>
<td>Select a certificate with a <code>Type</code> value of Java Key Store or PKCS12 Key Store to send to the web service provider. An administrator must set up a key store and generate certificates for the client and server.</td>
</tr>
</tbody>
</table>

### Enable mutual authentication

You can configure a SOAP or REST message for mutual authentication using a protocol profile.

**Role required:** `web_service_admin` or `admin`

1. Navigate to **System Web Services > SOAP Message** or **System Web Services > REST Message**.
2. Select a message record.
3. Select the `Use mutual authentication` check box.
4. In the `Protocol profile` field, select a protocol profile configured for mutual authentication.
5. Click **Update**.

### Outbound web services logging

Log requests and responses for outbound web services such as REST and SOAP.

Outbound request logging allows you to better understand what 3rd-party services your instance accesses and the volume of outbound requests. Additionally, logging can provide valuable information when debugging outbound integrations.

Outbound web services logging tracks outbound REST and SOAP requests, as well as outbound requests made using the GlideHTTPRequest and GlideHTTPClient APIs.

To view outbound web service logs, navigate to **System Logs > Outbound HTTP Requests**.

All log information is read only.

### Configure outbound logging

You can configure outbound request logging to log basic, elevated, or all HTTP request and response information for specific domains.

To configure the log level for a REST method or SOAP message function, navigate to the record you want to configure and click the **Set log level** related link, then select a log level for the current record.
To modify the log level for multiple outbound requests, navigate to System Web Services > HTTP Log Levels and change the log levels using the list. All outbound requests that have been configured with a specific log level are listed.

You can override the log level for all outbound requests using the properties glide.outbound_http_log.override and glide.outbound_http_log.override.level. Use these properties only for a limited time when troubleshooting.

You can set the log level in a script using the setLogLevel() function from the SOAPMessageV2 and RESTMessageV2 APIs. For more information about using these APIs, refer to the API documentation.

## Outbound request log levels

Certain elements are logged based on the configured log level.

### Logged elements

The following elements from the request and response are logged depending on the configured log level.

<table>
<thead>
<tr>
<th>Field</th>
<th>Basic</th>
<th>Elevated</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>HTTP Method</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Protocol</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Scheme</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Hostname</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Path</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>HTTP response status</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Request length</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Response length</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Total call time</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>System ID</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Source table</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Source record ID</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Scope</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Session</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

**Note:** You must have read access for records in the specified **Source table** to view this field.
<table>
<thead>
<tr>
<th>Field</th>
<th>Basic</th>
<th>Elevated</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>User</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>MID server</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Request headers</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Request query</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Response headers</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Request body</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Response body</td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

**Outbound request logging blacklist**

You can blacklist domains to allow only basic-level logging on those domains.

To blacklist a domain, navigate to **System Web Services > Outbound HTTP Log Domain Blacklist** and create a new record. You must have the web_service_admin role to blacklist domains.

**Outbound request logging blacklist domain requirements**

When you blacklist outbound request logging for a domain, the value you enter in the Domain field must meet certain requirements.

**Requirements**

The value you enter in the **Domain** field must meet the following requirements.

- Begins with `HTTP://` or `HTTPS://`.
- Is a domain pattern or IP address.
- Ends with alphanumeric characters preceded by a period, such as `.com`.
- Includes at most a single wildcard character immediately following the scheme and hierarchical portion of the domain pattern.

**Wildcard**

You can use a single wildcard character (*) in the domain pattern. Use this wildcard immediately following the scheme and hierarchical portion of the domain pattern, such as `http://*.domain.com` to include all subdomains. The wildcard must immediately follow the scheme and hierarchical portion of the domain pattern. If you use an IP address instead of domain pattern, you must enter the full IP address without a wildcard.

**Note:** You cannot use multiple wildcards, or specify a wildcard without a domain pattern. Values such as `*` or `*.*` are not supported.
Domain matching

When evaluating the Origin header in a request, ServiceNow prioritizes rules that match the domain pattern exactly. If no exact match is found, the next closest match is used.

For example, if there are rules for the domain patterns http://*.blog.mysite.com and http://*.mysite.com, a request from http://alice.blog.mysite.com will match the http://*.blog.mysite.com pattern.

Examples of valid and invalid domains

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid domain</td>
</tr>
<tr>
<td>Invalid domain</td>
</tr>
<tr>
<td>http://*.ms.net</td>
</tr>
<tr>
<td>https://*.ms.com</td>
</tr>
<tr>
<td><a href="http://192.168.1.1">http://192.168.1.1</a></td>
</tr>
<tr>
<td>http://*.service-now.com</td>
</tr>
<tr>
<td>http://*.com</td>
</tr>
</tbody>
</table>

Outbound web service logging properties

These properties allow you to control the behavior of outbound web service request logging.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>glide.outbound_http.db.log</td>
<td>By default, log information is written to a log file and to the instance database. Set this property to false to disable database logging.</td>
</tr>
<tr>
<td></td>
<td>• Type: true</td>
</tr>
<tr>
<td></td>
<td>• Default value: true</td>
</tr>
<tr>
<td></td>
<td>• Location: Add to the System Property (sys_properties) table</td>
</tr>
<tr>
<td>glide.outbound_http.text.content_types</td>
<td>A comma-separated list of content types. The body of requests or responses with one of these content types is logged. By default, the content types text/*, application/json, and application/xml are always logged. Use this property to add additional content types.</td>
</tr>
<tr>
<td></td>
<td>• Type: String</td>
</tr>
<tr>
<td></td>
<td>• Default value: none</td>
</tr>
<tr>
<td></td>
<td>• Location: Add to the System Property (sys_properties) table</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>glide.outbound_http.content.max_limit</td>
<td>The maximum number of characters logged from a request or response body. The first characters of the body, up to this limit, are logged. This property has a maximum possible value of 1000.</td>
</tr>
<tr>
<td></td>
<td>• Type: integer</td>
</tr>
<tr>
<td></td>
<td>• Default value: 100</td>
</tr>
<tr>
<td></td>
<td>• Location: Add to the System Property (sys_properties) table</td>
</tr>
<tr>
<td>glide.outbound_http_log.override</td>
<td>When this property is set to true, the property glide.outbound_http_log.override.level determines the log level for all requests and responses. This log level overrides any other log level settings. Only use this property for a limited time when troubleshooting.</td>
</tr>
<tr>
<td></td>
<td>• Type: true</td>
</tr>
<tr>
<td></td>
<td>• Default value: false</td>
</tr>
<tr>
<td></td>
<td>• Location: Add to the System Property (sys_properties) table</td>
</tr>
<tr>
<td>glide.outbound_http_log.override.level</td>
<td>The log level to use for all requests and responses when glide.outbound_http_log.override is true. Valid values are basic, elevated, and all.</td>
</tr>
<tr>
<td></td>
<td>• Type: String</td>
</tr>
<tr>
<td></td>
<td>• Default value: basic</td>
</tr>
<tr>
<td></td>
<td>• Location: Add to the System Property (sys_properties) table</td>
</tr>
</tbody>
</table>

Server Name Indication for outbound web services

Outbound HTTP requests, including outbound SOAP, REST, and GlideHTTPClient requests support Server Name Indication.

Server Name Indication (SNI) is a TLS extension that enables service providers to serve multiple host names from a single IP address. With SNI support, outbound web service requests specify the hostname to connect to as part of the TLS handshake. This information allows the server to present the correct certificate to the client.

SNI support is provided for outbound requests sent directly from an instance, as well as outbound requests sent through a MID Server. The hostname is automatically included in the HTTP request when SNI is enabled, it is not necessary to configure the request for SNI support.

For example, when SNI support is enabled, a REST message sent to the endpoint `https://myhost.com/some/path` will include the domain `myhost.com` in the TLS SNI extension.

Configure Server Name Indication

Server Name Indication support is disabled by default for all outbound HTTP requests.

To enable or disable SNI support for outbound HTTP requests sent directly from the instance, set the system property `glide.outbound.tls_sni.enabled` to true or false (default).
To enable or disable SNI support for outbound HTTP requests sent through a MID Server, create the property `glide.outbound.tls_sni.enabled` on the MID Server with a value of true or false (default), then restart the MID Server.

**Note:** After you change the property, wait at least 30 seconds before testing a request to ensure that a cached connection is not reused.

### HTTP Client Connection Management

Outbound HTTP(S) connections from a base system instance or inbound connections from MID Servers, the ODBC driver, and other clients are maintained and reused where possible. Connection pooling is used to keep track of HTTP(S) client connections to determine if they are alive and available for reuse.

ServiceNow HTTP client code means:

- Any application or script which makes outbound HTTP(S) requests from a base system instance.
- ServiceNow code in the MID Server or the ODBC driver which makes HTTP(S) requests to one or more base system instances.

**Note:** This discussion does not apply to browser-to-instance communication. No changes have been made with respect to the management of HTTP(S) connections for browser-based communication with ServiceNow. This discussion also does not apply to customer-developed Web Services clients making requests to ServiceNow.

### What Should the Customer Do?

Users should monitor performance, such as the decreased time for loading Discovery data and improved ODBC driver performance. For systems with an unusually large amount of simultaneous outbound HTTP(S) activity, such as numerous third-party integrations or high-volume automated activities which generate HTTP(S) requests from the base system instance to other places, review the `max_connections` and `max_connections_per_host` properties to ensure that the settings are sufficient. This enhancement has no impact on end-user connections from browsers and no impact on connections from customer-developed Web Services client applications.

### HTTP Connection Management Properties

Connection pooling is controlled by three properties. The default values for these properties are appropriate for most customers. The Glide properties are dynamic, meaning that changes to these properties will take effect immediately. No outage or restart is required to update the values.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>glide.http.use_connection_mgr</td>
<td>Switches connection pooling on and off. To disable the new behavior (not recommended) set glide.http.use_connection_mgr to false.</td>
<td>true</td>
</tr>
<tr>
<td>glide.http.connection_mgr.max_connections</td>
<td>Controls the total number of permitted HTTP(S) connections outbound from the base system</td>
<td>20</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Default Value</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>glide.http.connection_mgr.max_connections</td>
<td>Controls how many of the glide.http.connection_mgr.max_connections can communicate in parallel with any particular host. If the maximum setting for any of these values is reached during normal operations, a script or background thread may have to wait briefly to obtain a connection.</td>
<td>4</td>
</tr>
</tbody>
</table>

**ODBC driver**

The ServiceNow open database connectivity (ODBC) driver provides read-only access to the database associated with your ServiceNow instance.

The ODBC driver is compliant to version 3.52 of the Microsoft ODBC core API conformance. The ServiceNow ODBC driver uses ServiceNow web services support for a query-only interface. The ODBC driver supports only select statements or read-only functions and does not modify your instance data. Because the ODBC driver uses the web services interface, platform-wide access control (ACL) is enforced and data security is in place.

**Note:**

The ODBC driver has these limitations:

- The ODBC driver supports only select statements or read-only functions, and does not modify your instance data.
- There is no supported way to use the ODBC driver with a Java client application or with a Java JDBC-ODBC bridge.
- There is a hard-coded limit of 512 characters when accessing views through ODBC. Because of this limitation, a maximum of 11 table sys_ids can be included in an ODBC view query. Anything over 11 tables results in an error.

**Note:**

Versions of the ODBC driver older than 1.0.7.3 are no longer supported.

**Getting started with ODBC**

Before installing the ODBC driver, view the video on this page, create an ODBC user account, assign the odbc role, and define an ACL rule for the odbc role.

A ServiceNow user with the admin role can perform these procedures.

This video demonstrates how to install, configure, and test the ServiceNow ODBC driver, which provides read-only access to the database associated with your ServiceNow instance.

Before downloading and installing the ODBC driver, review the requirements to ensure that your configuration is compatible.
Create an ODBC user account and assign the odbc role

The ODBC driver communicates with your ServiceNow instance as a specific user. Create an ODBC user account and assign the odbc role to enable the user to communicate.

Role required: admin

The odbc role contains various additional roles, including the soap_query role required to make ODBC requests and the itil role required to access core tables such as the incident table. You can modify the odbc role to allow access to other tables as needed by adding additional child roles.

1. Navigate to User Administration > Users.
2. Click New.
3. In the User ID field, enter odbc.user.
4. Enter a Password for this user.
5. From the form context menu, select Save.
6. In the Roles related list, click Edit.
7. Use the slushbucket to add the odbc role, and then click Save.
8. Click Submit.

In a separate browser session, confirm that the odbc.user is able to log in to your ServiceNow instance.

Define an ACL rule for the odbc role

Define an ACL rule for the odbc role to provide read access to the incident table. You can create other ACL rules for the odbc role to provide read access to other tables.

Role required: admin

1. Elevate the session permissions to security_admin so you can create ACL rules.
3. Click New.
4. From the Operation choice list, select read.
5. From the Name choice list, select Incident (incident).
6. Leave the second Name choice list as None.
7. From the form context menu, select Save.
8. In the Requires role related list, click Edit.
9. Use the slushbucket to add the ODBC role, and then click Save.
10. Click Submit.

Installing the ODBC driver

Review setup requirements, download the ODBC driver installer, and install the ODBC driver to a computer.

You can install the ServiceNow ODBC driver on Microsoft Windows computers. To install the ODBC driver, set up an ODBC user in your ServiceNow instance, then download and install the ODBC driver. If you already have the ODBC driver installed, you can upgrade to the newest version.

ODBC driver installation requirements and supported software

Install the ServiceNow ODBC driver on Microsoft Windows computers.
# Installation requirements

Before installing the ODBC driver, ensure that your configuration meets these requirements.

## Installation requirements

<table>
<thead>
<tr>
<th>Category</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active user</td>
<td>The user record on the instance used to perform the queries.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The account used to connect to the instance via the ODBC driver must be defined on the instance. Accounts using single sign-on are not supported by the ODBC driver.</td>
</tr>
<tr>
<td>The soap_query role</td>
<td>The user you use to query the database must have the soap_query role if the instance uses the glide.soap.strict_security high security setting.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning:</strong> Do not enable WS-Security for all SOAP requests by setting the glide.soap.require_ws_security system property. It is incompatible with the ODBC driver. Enabling this setting blocks both ODBC driver and MID Server connections. Instead, use basic authentication.</td>
</tr>
<tr>
<td>Target Table ACLs</td>
<td>The user you use to query the database must have read access for the tables that you want to query. For example, a user with the itil role can read task tables, such as the incident table.</td>
</tr>
<tr>
<td>Target Table Web Service Access</td>
<td>The table you want to query must allow web service interaction. You can enable web service interaction using the application access settings.</td>
</tr>
<tr>
<td>Operating System</td>
<td>The ServiceNow ODBC driver supports installation on the following operating systems:</td>
</tr>
<tr>
<td></td>
<td>- Windows XP</td>
</tr>
<tr>
<td></td>
<td>- Windows Server 2003</td>
</tr>
<tr>
<td></td>
<td>- Windows Server 2008</td>
</tr>
<tr>
<td></td>
<td>- Windows Server 2012</td>
</tr>
<tr>
<td></td>
<td>- Windows Server 2016</td>
</tr>
<tr>
<td></td>
<td>- Windows Vista</td>
</tr>
<tr>
<td></td>
<td>- Windows 7</td>
</tr>
<tr>
<td></td>
<td>- Windows 8</td>
</tr>
<tr>
<td></td>
<td>- Windows 10</td>
</tr>
<tr>
<td>Category</td>
<td>Requirement</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Hardware                | • RAM: 1 GB minimum  
• Disk space: 135 MB for installation, 200 MB for writing cache files during usage |
| Account                 | The Windows account used for the installation must have local Administrator rights to install an ODBC driver.                              |
| Networking              | During usage, the ODBC driver requires HTTPS (port 443) connectivity to the ServiceNow instance. The communication between the ODBC driver and the ServiceNow instance uses standard SOAP web services. |
| End User License Agreement | Read the End User License Agreement for the ServiceNow ODBC driver.                                                                           |

For more information, see *Application access settings.*

**Supported software**

The following table lists the operating systems and reporting applications compatible with each version of the ODBC driver.
### Supported software

<table>
<thead>
<tr>
<th>Driver Version</th>
<th>Operating System</th>
<th>Microsoft Excel</th>
<th>Microsoft SQL Server</th>
<th>Crystal Reports</th>
<th>Tableau</th>
<th>Informatica</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.13 and later</td>
<td>Windows XP SP2</td>
<td>2007</td>
<td>2008</td>
<td>2008</td>
<td>8.1</td>
<td>The ODBC driver provides only basic support for Informatica. Use the ODBC driver with Informatica only for simple operations. Thoroughly test integrations with Informatica before using them in a production environment.</td>
</tr>
<tr>
<td></td>
<td>Windows Vista</td>
<td>2010</td>
<td>2012</td>
<td>2011</td>
<td>8.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows 7</td>
<td>2013</td>
<td>2014</td>
<td>2013</td>
<td>8.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows 8.x</td>
<td></td>
<td></td>
<td></td>
<td>9.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows Server 2003</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows Server 2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows Server 2008 R2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows Server 2012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows Server 2012 R2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows Server 2016</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Driver Version</td>
<td>Operating System</td>
<td>Microsoft Excel</td>
<td>Microsoft SQL Server</td>
<td>Crystal Reports</td>
<td>Tableau</td>
<td>Informatica</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------</td>
<td>------------------</td>
<td>----------------------</td>
<td>----------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>1.0.9 - 1.0.12</td>
<td>• Windows XP SP2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows Vista</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows 8.x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows Server 2003</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows Server 2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows Server 2008 R2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows Server 2012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows Server 2012 R2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0.8 and earlier</td>
<td>• Windows XP SP2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows Vista</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows 8.x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows Server 2003</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows Server 2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Windows Server 2008 R2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Download and install the ODBC driver

Download the ODBC driver from the ServiceNow Knowledge Base and install the driver for the first time.

You must have administrator-level access for the Windows computer onto which you want to install the ODBC driver.
You must have access to the ServiceNow Knowledge Base to download the ODBC driver. If you do not have access to the Knowledge Base, contact your ServiceNow administrator.

If this is the first time the driver is installed, the installer will be in first time installation mode and will prompt for the driver to be installed. Install only one version of the ODBC driver on a computer. If the ODBC driver was previously installed, the installer will be in upgrade mode and will prompt for removal of the previous driver first.

**Note:** Versions of the ODBC driver older than 1.0.7.3 are no longer supported.

1. To download the ODBC driver, navigate to [KB0540707](#).
2. Download the ODBC driver version compatible with your computer’s operating system and the application you are using to query the database.
   The ODBC driver is available as 32-bit or 64-bit. Most applications require the 32-bit ODBC driver even if the operating system is 64-bit.
3. Right-click the executable and select *Run as Administrator* to launch the installer.
   You are presented with the following InstallShield dialog box.

4. Click *Next*.
5. Read and accept the End User License Agreement.
6. Select the target directory for installing the ServiceNow ODBC driver.
   The default directory is `C:\Program Files\Service-now\ODBC`.
7. Specify the following parameters, which are required to create an ODBC data source that can be used to create a DSN.

- **Data Source Name**: A short name to identify this data source.
- **Description**: A short description of the driver. The driver's version number is appended at the end of this value.
- **Service Name**: The name that can be selected in the **Service Name** field of the ODBC Administrator.
- **Service Data Source**: The name that can be selected in the **Service Data Source** field of the ODBC Administrator.

Usually the default values are appropriate.

8. Select the **Program Folder** to create links for the driver. This is the program folder that appears under the **Start** menu.
The installation creates the following links in the menu.

- **Interactive SQL (ODBC)**: An interactive SQL command window for directly testing SQL statements.
- **Management Console**: A Microsoft MMC snap-in for configuring default properties for the ODBC driver.
- **ODBC Administrator**: A Microsoft ODBC Administrator program.

The driver code is copied to the target folder.
A progress bar appears.

9. When prompted, click Finish to complete the installation.

Upgrade the ODBC driver

If you have previously installed an older version of the ODBC driver, run the installer to uninstall the previous version, and then run the installer again to upgrade.

To check the build date and time of the ODBC driver, use CheckVersion located in the ServiceNow\ODBC\ip\tools folder. This is an executable Windows host script that reports the build date and time of the current ODBC driver. Use it to assist ServiceNow Technical Support to determine which build of the ODBC driver is running. If the CheckVersion tool is absent, the ODBC driver is out of date; upgrade to the current version. To check the version of an older ODBC driver, see the previous version information.

Note: The ODBC installation also has a ServiceNow\ODBC\tools folder, which is not the correct path for the CheckVersion tool.

1. Right-click the executable and select Run as Administrator.
2. Click **OK** when prompted to uninstall the current driver, which is required for the upgrade. A list appears, displaying the existing ODBC DSN names that you have previously created. You have the option to delete them.

3. Select **Yes** to remove all previous DSNs or **No** to keep them for use with the upgraded driver. An ODBC DSN is a connection **handle** to use the ODBC driver in an application. For more information from Microsoft, see:

4. After removing the previous ODBC driver, double-click the executable again to run the installer. Then, follow the steps in Download and install the ODBC driver. If you encounter errors when uninstalling the ODBC driver, refer to the troubleshooting uninstalling ODBC knowledge article.

After installing the ODBC driver, configure it to connect to your ServiceNow instance.

**Configuring ODBC**

After installing the ODBC driver, configure it to connect to your ServiceNow instance and to communicate through a proxy server, if applicable, and set properties to control ODBC behavior.

**Configure the ODBC driver**

Configure the ODBC driver to connect to your ServiceNow instance.

You must have administrator-level access for the Windows computer on which you want to configure the ODBC driver.

After the driver is installed, configure it for your instance. The driver is preconfigured to connect to https://demoodbc.service-now.com using the DSN ServiceNow. There are two ways to configure connectivity for the driver:

- Configure the global default used by all newly created DSNs.
Configure each new DSN with its own connection.

1. In Windows, navigate to Start > Programs > ServiceNow ODBC Management Console.
2. Expand the Console Root tree to: ServiceNow ODBC Manager\Manager\<installation location>\Services\ServiceNow_ODBC\Data Source Settings\ServiceNow\IP Parameters.
3. Double-click the DataSourceIPProperties attribute.
4. Change the Value to the URL of your instance, such as https://<instance>.service-now.com.
   If integrating the ODBC driver with Edge Encryption, change the Value to the URL of your encryption proxy. See Integrate Edge Encryption with the ODBC driver for more information.
5. Click OK.

Configure the global DSN default

Configure the global default used by all newly created DSNs.

You must have administrator-level access for the Windows computer on which you want to configure the global DSN default.

A default DSN is preloaded with the ODBC driver installation ServiceNow data source. This preloaded DSN connects using the default connection URL, which is set to https://demo.service-now.com. To change the global default for the instance URL, do the following.

1. In Windows, navigate to Start > Programs > ServiceNow ODBC > Management Console.
2. Expand the Console Root tree using the following path:
   OpenAccess SDK 6.0 Manager\<installation location>\Services\ServiceNow_ODBC \Data Source Settings\ServiceNow\IP Parameters
3. Double-click the DataSourceIPProperties attribute for the ServiceNow data source setting to open the Properties dialog box.
4. Change the value to the URL of your instance, using the following format, and then click **OK**: 

   https://<your instance>.service-now.com

Create a new DSN

Use the ODBC driver and the ServiceNow data source to create an unlimited number of DSNs configured to connect with different instance URLs.

You must have administrator-level access for the Windows computer on which you want to create a new DSN.

Select the target instance for your ODBC connection by DSN name. As an option during installation or upgrade, you can elect to keep the DSNs when you uninstall.

Instead of creating a new DSN, you can specify a connection string to connect with different instance URLs.

1. In Windows, navigate to **Start > Programs > Service-now ODBC > ODBC Administrator**.
2. To create a system DSN, select the **System DSN** tab, and then click **Add**.
3. Select ServiceNow **ODBC driver 32-bit** from the list, and then click **Finish**.
4. Configure the driver and its connection URL by specifying the `url=` parameter value in the **Custom Properties** field. For example:

   ```
   url=https://myinstance.service-now.com
   ```
5. Click OK.

You can now use the new driver.

**Configure the logging level of the ODBC driver**

Change the logging level of the ODBC driver.

You must have administrator-level access for the Windows computer on which you want to configure the logging level.

1. In Windows, navigate to Start > Programs > ServiceNow ODBC > Management Console.
2. (ODBC version 1.0.8) Within the management console, navigate to <your_installation_directory> > Services > ServiceNow_ODBC > Service Settings > IP Parameters.
3. (ODBC version 1.0.8) Change the value of the ServiceJVMOptions attribute to the desired logging level.
4. Within the management console, navigate to `<your_installation_directory> > Services > ServiceNow_ODBC > Service Settings > Logging.
5. Change the value of the `ServiceDebugLogLevel` by selecting all available check boxes.

6. In Windows, navigate to Start > Programs > ServiceNow ODBC > ODBC Administrator.
7. In the ODBC Administrator, select the **Tracing** tab.
8. Navigate to the path in the **Log File Path** field and delete the old log file, if it exists.
9. Click **Start Tracing Now**.
10. Enable SOAP debugging for your ServiceNow instance.

**Configure the ODBC driver for large data sets**

You can set two ODBC driver properties to deal with errors you receive when using queries that return large amounts of data.

You must have administrator-level access for the Windows computer on which you want to configure the ODBC driver for large data sets.

These properties are set using the ODBC Management Console that is available on Windows operating systems. For more information, see [ODBC management console properties](#).

1. In Windows, navigate to **Start** > **Programs** > **ServiceNow ODBC** > **Management Console**.
2. Navigate to **Data Source Settings** > **ServiceNow** > **IP Parameters**.
3. Set the **Timeout** property to be more than the glide.soap.request_processing_timeout.odbc value.
4. Navigate to **Services** > **Service Settings** > **SQL Engine Parameters**.
5. Increase the **ServiceSQLDiskCacheMaxSize** property.

Typically, when running a query that returns 50,000 rows, the default value of 200 must be increased.

**Configure ODBC to use proxy servers**

The ODBC driver can be configured to route its HTTP SOAP requests via an HTTP proxy server.

You must have administrator-level access for the Windows computer on which you want to configure ODBC to use proxy servers.

Setting up a proxy server gives you the option to control access to the ServiceNow instance from the proxy server, and potentially allows a network configuration that can monitor usage statistics. However, because the proxy server intercepts the ODBC driver's requests to your ServiceNow instance, it will degrade the performance of the driver.

**Note:** This feature is recommended for use with ODBC driver builds dated 7/15/2011 or later.

To enable the use of proxy servers, the custom properties for proxy server settings must be defined first for the data source. After that, these properties can be overridden by specific ODBC DSNs.

1. In Windows, navigate to **Start** > **Programs** > **ServiceNow ODBC** > **Management Console**.
2. Expand the Console Root tree to: **ServiceNow ODBC Manager\Manager\<installation location>\Services\ServiceNow_ODBC\Data Source Settings\ServiceNow\IP Parameters**.
3. Double-click the **DataSourceIPProperties** attribute.
ODBC proxy

4. Set the following custom properties to configure the ODBC proxy server.

**Configure ODBC to use proxy servers**

<table>
<thead>
<tr>
<th>Property name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>proxy_host</td>
<td>The proxy server host name or IP address.</td>
<td>proxy.company.com</td>
</tr>
<tr>
<td>proxy_port</td>
<td>The proxy server port number.</td>
<td>8080</td>
</tr>
<tr>
<td>proxy_user_name</td>
<td>The proxy server user name or id, used in an authenticating proxy configuration.</td>
<td>odbc_user</td>
</tr>
</tbody>
</table>
Setting ODBC properties

The following properties customize connectivity and optimize the query behavior of the ODBC driver.

ODBC administrator properties

These properties are specified in the ODBC Data Source Administrator for the DSN or in the Custom Properties field of the login dialog box.

ODBC administrator properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchSize</td>
<td>During fetching of results from the instance, this batch size configures the number of records to fetch for every request. Typically, the default is an optimal number for normal sized rows. If an error occurs during fetching of records that indicates this value should be lowered, you can modify it to optimize memory usage versus performance.</td>
<td>2000</td>
</tr>
<tr>
<td>url</td>
<td>This is the ServiceNow instance URL or endpoint. It should indicate the URL to the ServiceNow instance you want to connect to.</td>
<td><a href="https://demo.service-now.com">https://demo.service-now.com</a></td>
</tr>
<tr>
<td>EnablePassThrough</td>
<td>During processing of aggregate functions, enabling pass through mode allows directly calling Aggregate Web Service for optimized and speedy response. Whenever possible, this mode should be left enabled.</td>
<td>true</td>
</tr>
<tr>
<td>Property Name</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>debug</td>
<td>By default, debugging messages are not produced. Set <strong>debug</strong> to <strong>true</strong> when you operate the ODBC driver from the ISQL console window to write all HTTP-related network communication traffic to the console window. When using this option, set <strong>gzip</strong> to <strong>false</strong> so that data is not compressed. Otherwise, the data is unreadable.</td>
<td>false</td>
</tr>
<tr>
<td>gzip</td>
<td>By default, data sent over the network is compressed. Set <strong>gzip</strong> to <strong>false</strong> when using the <strong>debug</strong> parameter to write network communication to the ISQL console so that data is not compressed.</td>
<td>true</td>
</tr>
<tr>
<td>timeout</td>
<td>Specifies the socket inactivity timeout value in seconds.</td>
<td>175</td>
</tr>
<tr>
<td>retries</td>
<td>Number of times to retry the failing request in the event of a socket timeout error.</td>
<td>0</td>
</tr>
<tr>
<td>mode</td>
<td>The query mode used to parse complex where clauses. You can configure the ODBC driver query mode to use either AND or OR operators.</td>
<td>or</td>
</tr>
<tr>
<td>EnableDBSchema</td>
<td>The ODBC driver issues a database schema request to retrieve table names from the instance. This functionality is enabled by default so reporting applications such as Microsoft Excel can display a list of tables to query from. Disabling this property may improve the performance of the first query sent from a reporting application, especially if the instance has a large number of tables.</td>
<td>true</td>
</tr>
</tbody>
</table>
ExtendedSchemaCache

The ODBC driver caches the database schema for each connection. When a new connection is created, the driver clears the database cache and queries the database schema from the instance again. This behavior is beneficial when connecting to different data source, or when modifying the table schema. When querying a single data source with a consistent schema, enable this property to avoid sending unnecessary schema requests, including when EnableDBSchema is true. default: false

LegacyDurationTimeZone

The ODBC driver returns timer and duration field values in the UTC timezone by default, starting with the 1.0.10 version. When this property is true, the ODBC driver returns timer and duration field values using the display value, as shown in the UI. This property can be used to preserve compatibility with legacy integrations that depend on the display value. See KB0583982 for details about this behavior. default: false

If you need to use more than one of these properties in your connection, concatenate the settings with a semicolon (;) delimiter. For example, the following string sets the URL to a specific instance and changes the batch size to 200 records.

url=https://demo1234.service-now.com;BatchSize=200

**ODBC management console properties**

You can access these properties from the ODBC Management Console available in the Windows Start menu at ServiceNow ODBC > Management Console.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceJVMOptions</td>
<td>JVM command line properties and option. For example, to change the maximum Java heap size, modify the -Xmx150m parameter.</td>
<td>-Xms64m -Xmx150m</td>
</tr>
</tbody>
</table>

© 2020 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
### ServiceJVM options

You can specify these values within the ServiceJVMOptions parameter in addition to standard JVM arguments such as `-Xmx`.

#### Service JVM options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-DLOG_FILE_NAME</td>
<td>The location of the ODBC log file. This property is available starting with the ODBC driver 1.0.7.1 release.</td>
<td><code>${user.home}\AppData\Local\ServiceNow\odbc\logging\odbc.log</code></td>
</tr>
<tr>
<td>-DLOG_LEVEL</td>
<td>The logging level used when writing to the ODBC log file. You can specify the logging level using Logback levels, such as TRACE, INFO, or ERROR. This property is available starting with the ODBC driver 1.0.8 release.</td>
<td>INFO</td>
</tr>
</tbody>
</table>

### Instance properties

An administrator can configure these properties by adding a property or modifying an existing one in the ServiceNow instance.
Instance properties

<table>
<thead>
<tr>
<th>Property name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>glide.db.max.aggregates</td>
<td>The maximum number of rows returned by aggregate functions.</td>
<td>100000</td>
</tr>
<tr>
<td>glide.db.max_view_records</td>
<td>The maximum number of rows returned by a database view.</td>
<td>10000</td>
</tr>
</tbody>
</table>

Test the ODBC driver

After configuring the ODBC driver, test that the driver can connect to the base system instance as the ODBC user and can query data from a target table.

You must have administrator-level access for the Windows computer on which you want to test the ODBC driver.

To test the connection, run the ODBC Administrator program.

1. In Windows, navigate to Start > Programs > ServiceNow ODBC > ODBC Administrator. The ServiceNow ODBC data source is installed as a system data source.
2. Select the System DSN tab, and then select the ServiceNow data source.
3. Click Configure.
4. Click **Test Connect** in the ODBC driver Setup dialog box.
5. Enter the login credentials.

These are the usual base system login credentials for the ODBC user you created.
6. Click **OK** to log in to the data source.
7. Click **OK** again when the success message appears.

**Enable debug logging**

If you experience unexpected behavior when using the ODBC driver, you can enable debug logging and generate debug logs to help identify the issue.

Debug logs can be useful when submitting an incident with ServiceNow Technical Support.

When you enable debug logging, note the version and bitness (32 bit or 64 bit) of the installed ODBC driver, the Windows operating system, and the client application you are using with the ODBC driver.

To generate debug logs, follow these steps.

1. Close all active applications that may use the ODBC driver.
2. Navigate to one of these paths, based on your operating system.
   - For Windows 7: `C:\Users\<user_name>\AppData\Local\ServiceNow\odbc\logging`
• For Windows XP and earlier: C:\Program Files\ServiceNow\ODBC\%LOCALAPPDATA%\ServiceNow\odbc\logging

3. Delete any existing log data to ensure that you log only relevant information.
4. Run a query that produces the unexpected behavior, then immediately close the application and review the log files.

Test a query

To verify that the user has the appropriate permissions to send requests to the instance using ODBC, run a query using Interactive SQL.

For testing, use a query that returns exactly one record, such as a query using the Number value of a record.

1. In the base system instance, navigate to Incident > All.
2. Record the Number of an incident record.
3. On the computer where the ODBC driver is installed, navigate to Start > Programs > ServiceNow ODBC > Interactive SQL.
4. Enter connect "odbc.user"="password"@ServiceNow and press Enter.
5. Enter the following text, substituting the incident number you recorded: select short_description from incident where number='<incident number>';
6. Press Enter.

The instance should respond with the short description of the incident record.

ODBC troubleshooting

Review these troubleshooting resources to resolve issues with the ODBC driver.

For troubleshooting information, see the Knowledge Base articles troubleshooting ODBC driver issues and troubleshooting common ODBC error messages.

ODBC behavior

After testing the ODBC driver you can use it to query your instance database from a variety of client applications.

ODBC aggregate functions

The ODBC driver attempts to download data and apply aggregate functions locally. The ODBC driver supports the following aggregate functions.

• COUNT
• SUM
• MIN
• MAX
• AVG

Activate the Aggregate Web Service plugin to improve the performance of aggregate queries through the ODBC driver.
ODBC date and time values

The instance and the machine on which the ODBC driver is installed may use two different time zones. Date and time values returned by the ODBC driver are in the local time zone of the application using the driver, not the ServiceNow instance time zone.

Ensure that you query in accurate time zones for both the instance and the machine that hosts the ODBC driver. GlideRecord performs filtering based on the instance time zone, and the ODBC client is filtered based on the Windows time zone.

For example, an instance is in Central Standard Time (CST), and the ODBC driver is installed on a machine that is in Pacific Standard Time (PST). An incident is created on the instance at 2014-05-20 10:00:00, and the time that the incident was created is displayed in the UI as 10:00:00 for users in both time zones. However, in order to successfully query this incident by creation date and time, a user on the machine in PST must query 2014-05-20 08:00:00 instead of 2014-05-20 10:00:00.

Duration and timer type fields are returned using the UTC timezone, starting with ODBC version 1.0.10. See KB0583982 for details about this change.

ODBC display values

Some examples of how to use and work with ODBC display values are shown below.

- **Display values in Choice and Reference columns:**

  When querying a column of type Choice, Reference, Duration, or Timer, an additional column with the prefix dv_ is available that contains the display value. For example, you can select dv_caller_id to return the sys_user.name display value of the reference field from an incident record without making another request to the sys_user table.

  ![Example SQL](https://example.com/example_sql.png)

  Return the display value

- **Display values in filter conditions:**

  Display values can also be used in a filter condition. The ODBC driver optimizes the query condition and processes the filter on the server, for example, querying on the display value of sys_user for the caller_id field of an incident by using the dv_caller_id field name.
Display values in filter conditions

- Display values in aggregate queries:

  Aggregate queries can also take advantage of display values if you specify them in the group by or where clause, for example, grouping on the `caller_id` field of an incident, as well as specifying a filter for it. The query is optimized by passing through to the server.

```sql
SELECT COUNT(*), do_caller_id FROM incident WHERE do_caller_id IS NOT NULL GROUP BY do_caller_id;
```

<table>
<thead>
<tr>
<th></th>
<th>do_caller_id</th>
<th>caller_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>5137153cc611227c000bd1bd8cd2005</td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>5137153cc611227c000bd1bd8cd2005</td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>5137153cc611227c000bd1bd8cd2005</td>
<td></td>
</tr>
</tbody>
</table>

Display values in aggregate queries

**Querying table and column names**

You can get a list of accessible tables and columns based on the read ACLs for the querying user.

- The following query will return the names of all tables for which the querying user has read access:

  ```sql
  SELECT * FROM oa_tables;
  ```

- After you know the name of the table you want to query, you can query the names of all columns for which the user has read access. The querying user must have read access for both the table and the columns.

  ```sql
  SELECT * FROM oa_columns WHERE table_name='table_name';
  ```
**Note:** The `oa_tables` and `oa_columns` tables are internal ODBC tables. These tables are accessible only via the ODBC driver.

### Increase the field length in SQL queries

The ODBC driver limits the field length in SQL queries to the maximum length defined by the ServiceNow dictionary entry. You can increase the maximum field length to avoid truncating data.

**Role required:** admin

If the data coming from the ODBC source exceeds the field size of the dictionary entry, ServiceNow truncates the query output to fit the field size. If your data is truncated, you can do the following.

1. Increase the maximum length in the dictionary entry for the field in question.
2. Reconnect the ODBC driver to pick up the change.

**Note:** By default, the ODBC driver uses the VARCHAR data type to store query string output. When strings become very large (roughly 16000 characters), the ODBC driver uses the LONGVARCHAR data type instead. It is important to keep in mind, however, that the LONGVARCHAR data type has a more limited set of SQL commands that can be executed on it. For example, it does not support queries using scalar data.

### ODBC and client applications

See the following pages for examples of how to use the ODBC driver to create data sources from other applications.

### Use Interactive SQL with ODBC

Run the Interactive SQL application for quick verification of connectivity and to test query results without using a full application.

1. In Windows, navigate to **Start > Programs > ServiceNow ODBC > Interactive SQL (ODBC)**.
2. Enter the following command to connect to the base system instance. Select the appropriate user credentials in the format: `ID*password@DSNName`. The password cannot contain special characters.

   ```
   CONNECT odbcuser*password@ServiceNow
   ```
3. Issue a `SELECT` SQL command, such as:

```
SELECT NUMBER, short_description FROM incident;
```

Make sure to include the semicolon at the end of your query statement. You will be presented with a 'Cont>' prompt otherwise.
Specify the maximum number of rows returned

By default, ServiceNow only returns 100 rows of data with each iSQL query. If you need to return more rows of data, set the maxrows parameter for the iSQL session.

To return all rows set `maxrows` to zero:

```
maxrows 0
```

To return more than 100 rows set `maxrows` to a higher value. For example, to return 500 rows:

```
maxrows 500
```

**Note:** If running the Interactive SQL console from a shortcut, you must modify the shortcut Target to include the `-maxrows` parameter with the desired value.
SQL support
The ODBC driver embeds a third party SQL/ODBC engine from DataDirect, a division of Progress Software.

See the DataDirect SQL Reference for information on proper SQL syntax.

Note: The ServiceNow ODBC driver only supports SELECT statements. The driver ignores other SQL statements such as CREATE and ALTER.
ODBC driver in SQL Server

Use the ServiceNow ODBC driver in SQL Server as a Linked Server.

Using the ODBC driver in SQL Server as a Linked Server allows SQL Server to query tables from a ServiceNow instance directly via the ODBC driver. Only use the procedures described with SQL Server 2008 and 2012. Other versions of SQL Server may cause unexpected behavior. If you encounter unexpected behavior, refer to the troubleshooting linked server Knowledge Base article.

Required Permissions

Additional information on the required permissions for SQL Server Linked Servers can be found on the MSDN blog.

**Note:** Review this information if you encounter permission errors with SQL Server.

**ODBC SQL Server video tutorials**

Watch video tutorials about configuring and troubleshooting the ODBC driver with a SQL Linked Server.

Configuring Microsoft SQL Linked Server with the ODBC driver

**Troubleshooting Microsoft SQL Linked Server permissions**

Configure SQL Server

The following example configuration was performed on SQL Server 2008, installed on Windows Server 2008.

The ODBC Driver must be installed on the same computer on which SQL Server is installed.

1. Right-click the SQL Server Management Studio application and select as Run as Administrator.
2. Log in to the database to which you want to link.
3. Right-click Server Objects > Linked Servers.
4. Click New Linked Server.
5. Enter the following values in the dialog.
   - **Linked server:** SERVICENOW. This is the name of the Linked Server.
   - **Provider:** Microsoft OLE DB Provider for ODBC drivers
   - **Product name:** ServiceNow. This is an identifier. Enter any value that is appropriate.
   - **Data source:** ServiceNow. This is the name of your DSN.
6. Select **Security** from the **Select a page** list, and then enter the following security values:

1. For a login connection, select **Be made using this security context**.
2. Enter the user name and password for connecting to the ServiceNow instance.
3. Click **OK**.
7. Navigate to **Server Objects > Linked Server > Providers** and double-click Microsoft OLE DB Provider for ODBC drivers.
8. Select the following options.
   - Nested Queries
   - Level zero only
   - Support 'Like' operator

Tip:
ServiceNow recommends running the third-party provider in the out-of-process mode setting (`AllowInProcess=FALSE`). If you run the provider in-process (within the same process as SQL Server), then any issues with the provider can affect the SQL Server process, which in turn could result in crashing SQL server.
9. Test your connection by selecting the newly created linked server **SERVICENOW** and selecting **Test connection**.
10. Execute the following query in a query builder window to retrieve some results.

```
SELECT * FROM OPENQUERY (SERVICENOW , 'select Cast(sys_mod_count as Decimal(38,0)), number, short_description from incident' )
```

Number Precision Errors

You may encounter precision errors querying for decimal or number field values using the `OPENQUERY` syntax with the ODBC driver. In this case, use the `Cast` syntax to convert the precision. For example:

```
SELECT * FROM OPENQUERY (SERVICENOW , 'select Cast(sys_mod_count as Decimal(38,0)), number, short_description from incident' )
```

SQL Server Connection String
To use the ODBC driver directly in SQL Server 2008, specify the connection string in the following format.

Dsn=ServiceNow;uid =username;pwd =password

**Note:** The latest SQL Server 2008 patches are required for the ability to specify a connection string in the user interface, via the SQL import wizard.

**Using sp_addlinkedserver**

The following example creates a linked server named “ServiceNow ODBC” that uses the Microsoft OLE DB Provider for ODBC (MSDASQL) and the data_source parameter.

```sql
EXEC sp_addlinkedserver
    @server  = N 'ServiceNow ODBC' ,
    @srvproduct  = N '' ,
    @provider  = N 'MSDASQL' ,
    @datasrc  = N 'ServiceNow';
GO
```

After creating the linked server, you must update its properties to specify the login credentials.

**Use the ODBC driver in Excel**

After installing the ODBC driver and its associated DSN, use it in Excel as a data source provider.

1. In Excel open the **Data** tab.
2. Under **From Other Sources** open **From Microsoft Query**.
3. Select **ServiceNow** as your database (the default DSN name).
4. Clear the **Use the Query Wizard to create/edit queries** check box.

**Note:** The Excel Query Wizard does not support the listing of columns from a table name that contain an underscore ( _ ). Clearing this check box uses the Query Builder instead, which supports the use of this character.
5. Supply the ServiceNow user name and password.

6. Select a table from the ServiceNow instance and click Add.

7. Close the dialog box.

8. Select the table columns from which the Query Builder will retrieve data. Use the list above the table, or type the names directly into the columns, and then press Enter.

9. To retrieve the data and create the Excel record, click the Return Data icon or select File > Return Data to Microsoft Office Excel.
<table>
<thead>
<tr>
<th>number</th>
<th>short_description</th>
<th>dv_assigned_to</th>
<th>dv_state</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>INC000009</td>
<td>Reset my password</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000010</td>
<td>Need Oracle 10G/F2 installed</td>
<td>Don Goodlife</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000011</td>
<td>Need new Blackberry setup</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000012</td>
<td>eFax is not working</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000013</td>
<td>EMAIL is slow when an attach</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000014</td>
<td>missing my home directory</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000015</td>
<td>I can't launch my game anymore</td>
<td>Don Goodlife</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000016</td>
<td>Rain is leaking on main DNS</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000017</td>
<td>How do I create a sub-folder</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000018</td>
<td>Sales forecast spreadsheet is blank</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000019</td>
<td>Can't launch XWin32</td>
<td>Bud Richman</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000020</td>
<td>Request for a Blackberry</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000021</td>
<td>New employee hire</td>
<td>Beth Anglin</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000024</td>
<td>Issue with a web page</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000025</td>
<td>I need more memory</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000026</td>
<td>Seem to have an issue with my</td>
<td>Don Goodlife</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000028</td>
<td>My disk is still having issues.</td>
<td>Don Goodlife</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000027</td>
<td>please remove this host</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000029</td>
<td>I can't get my weather report</td>
<td>Don Goodlife</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000030</td>
<td>Lost connection to the wireless</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000031</td>
<td>EMAIL Server Down</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000032</td>
<td>EMAIL Server Down Again</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000033</td>
<td>File Server is 80% full - Needs</td>
<td>Don Goodlife</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000034</td>
<td>Does not look like a backup</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000035</td>
<td>Reset my password</td>
<td>Luke Wilson</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000036</td>
<td>Issue with networking</td>
<td>Luke Wilson</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC000037</td>
<td>Request for a new service</td>
<td>Howard Johnson</td>
<td>Open</td>
<td>1</td>
</tr>
</tbody>
</table>
The requested data is brought into Excel.
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>short description</td>
<td>dv assigned to</td>
<td>dv state</td>
<td>state</td>
</tr>
<tr>
<td>INC0000009</td>
<td>Reset my password</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000010</td>
<td>Need Oracle 10GR2 installed</td>
<td>Don Goodlife</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000011</td>
<td>Need new Blackberry setup</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000012</td>
<td>eFax is not working</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000013</td>
<td>EMAIL is slow when an attachment is involved</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000014</td>
<td>missing my home directory</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000015</td>
<td>I can't launch my game anymore</td>
<td>Don Goodlife</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000016</td>
<td>Rain is leaking on main DNS Server</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000017</td>
<td>How do I create a sub-folder</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000018</td>
<td>Sales forecast spreadsheet is READ ONLY</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000019</td>
<td>Can't launch X-Win32</td>
<td>Bud Richman</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000020</td>
<td>Request for a Blackberry</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000021</td>
<td>New employee hire</td>
<td>Beth Anglin</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000022</td>
<td>Issue with a web page</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000023</td>
<td>I need more memory</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000024</td>
<td>Seem to have an issue with my harddrive...</td>
<td>Don Goodlife</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000028</td>
<td>My disk is still having issues, can't delete file</td>
<td>Don Goodlife</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000027</td>
<td>please remove this hotfix</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000029</td>
<td>I can't get my weather report</td>
<td>Don Goodlife</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000030</td>
<td>Lost connection to the wireless network</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000031</td>
<td>EMAIL Server Down</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000032</td>
<td>EMAIL Server Down Again</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
</tbody>
</table>
Use the ODBC driver in Crystal Reports

After installing the ODBC driver and its associated DSN, use it in Crystal Reports as a data source provider.

**Note:** Crystal Reports includes the configuration file CRConfig.xml that contains the JVM minimum heap size (Xms) and maximum heap size (Xmx) values. When configuring the ODBC driver with Crystal Reports, ensure the ODBC driver uses the same minimum and maximum JVM heap size as Crystal Reports. If these values do not match, update the ODBC driver settings, not the Crystal Reports settings.

1. Create a new Standard Report

2. Create a new connection using the ServiceNow DSN
3. Select a table from the list of available tables
4. Select the available fields from the selected table
5. Click **Finish** to render the report
<table>
<thead>
<tr>
<th>number</th>
<th>short description</th>
<th>or Assigned To</th>
<th>or state</th>
<th>stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>INC0000001</td>
<td>Reset my password</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000009</td>
<td>Need Oracle 10G R3 installer</td>
<td>Don Goodf</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000011</td>
<td>Need new Blackberry setup</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000012</td>
<td>New email address</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000013</td>
<td>E-mail is being sent as CC</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000014</td>
<td>Missing my home directory</td>
<td>Open</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INC0000015</td>
<td>Can't launch my game offline</td>
<td>Open</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INC0000016</td>
<td>Rain is leaking on main 2 NS</td>
<td>Open</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INC0000017</td>
<td>How do I create a sub-folder</td>
<td>Open</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INC0000018</td>
<td>Sales forecast spreadsheet</td>
<td>Open</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INC0000019</td>
<td>Can’t launch X-WinS2</td>
<td>Open</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INC0000020</td>
<td>Request for a BlackBerry</td>
<td>Open</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INC0000021</td>
<td>New employee file</td>
<td>Beth Angel</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000022</td>
<td>Issue with a web page</td>
<td>ITIL User</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000023</td>
<td>Need more memory</td>
<td>Open</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INC0000024</td>
<td>Need to replace issue with</td>
<td>Don Goodf</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000025</td>
<td>My disk is still having issues</td>
<td>Don Goodf</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000026</td>
<td>Please remove this notice</td>
<td>System Administrator</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000027</td>
<td>I can’t get my weather report</td>
<td>Open</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INC0000028</td>
<td>Lost connection to the whole</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>INC0000029</td>
<td>EMAIL Server Errors</td>
<td>Open</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INC0000030</td>
<td>EMAIL Server Errors Again</td>
<td>David Loo</td>
<td>Open</td>
<td>1</td>
</tr>
</tbody>
</table>
Domain separation and ODBC driver

This is an overview of domain separation and ODBC drivers. Domain separation allows you to separate data, processes, and administrative tasks into logical groupings called domains. You can then control several aspects of this separation, including which users can see and access data.

Overview

Support: Data only

Domain separation in this application is supported at the Data only level, meaning it supports the data security model of separating visibility of data from one domain to another. To learn more, see Application support for domain separation.

Domain separation in Web Services

This is an overview of domain separation in Web Services. Domain separation allows you to separate data, processes, and administrative tasks into logical groupings called domains. You can then control several aspects of this separation, including which users can see and access data.

Overview

Support: Data only

Domain separation in this application is supported at the Data only level, meaning it supports the data security model of separating visibility of data from one domain to another. To learn more, see Application support for domain separation.
Index

Special Characters

.net 2259

A

AbstractDBObject
isValid() 375
AbstractDBObject API 375
AbstractReconciler
getReconciliationField() 376
getReconciliationKey() 375
hasChanged() 376
markAbsent() 377
markPresent() 377
process() 377
readDatabaseFields() 378
readDiscovered() 378
resolveReferenceFields() 378
setDatabaseFields() 379
setup() 379
track(String sys_id) 379
AbstractReconciler API 375
AbstractScriptProcessor
initialize(String request, String response, String process) 380
process() 380
AbstractScriptProcessor API 380
AbstractTimelineSchedulePage
add() 381
addSeparator() 382
elementMoveX() 382
elementMoveXY() 383
elementMoveY() 383
elementSuccessor() 384
elementTimeAdjustEnd() 385
elementTimeAdjustStart() 384
getItems() 385
inputBox() 386
itemMove() 386
setDoReRenderTimeline() 386
setPageTitle() 387
setStatusError() 387
setStatusPrompt() 387
setStatusSuccess() 389
access
enforcing 57
to Update Set picker 334
workflow scratchpad
from business rules 1969
access control rules
ACL rules 56, 56
ActivityHistoryRecord
addArrivedActivityToJoin() 389
addJoinFromActivityIds() 390
addTransition() 390
debugDump() 390
doesJoinContainActivity() 391
getLogString() 391
getSatisfiedJoinActivities() 391
getTransitionCount() 392
getUnSatisfiedJoinActivities() 392
isARollback() 392
isADestination() 393
isJoin() 393
isJoinSatisfied() 393
isJoinWaitingForActivity() 394
isRolledBack() 394
isTurnstile() 394
add role to every user 1972
add test step 102
adding
elements
scrolling 2284
field
service catalog checkout 1970
news
scroll 2286
news panel
scrolling 2285, 2286
scroll
news 2286
scrolling
elements 2284
news panel 2285, 2286
Agent Intelligence
GET 1594
AgentMetrics
AgentMetrics() 395
handleCounter(String metric, String name, String midId) 395
handleMemory(String metric, String name, String midId) 396
handleRGR(String metric, String name, String midId) 396
handleScalar(String metric, String name, String midId) 397
updateMetric(String tableName, String values, String metric, String midId) 397
AgentMetrics API 395
Aggregate API
GET 1597
AJAX 1919
AJAXClientTiming 1923
APCHandler
classifyAndIdentify() 398
APCHandler API 398
ApcPduOutletReconciler
getReconciliationField() 398
getReconciliationKey() 399
hasChanged() 399
readDatabaseFields() 399
readDiscovered() 400
setDatabaseFields() 400
ApcPduOutletReconciler API 398

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
APIs

Glide Server 1941
application
import from source control 258
link to source control 256
application administration 56, 56
Application Administration ACL Inheritance
sys_scoped_admin_acl_inheritance 56
application development 79
application files
adding 245
searching
by name 247
by record values 247
by type 247
application scope 7
applications
access settings
design time
denying 33
granting 35
runtime
default permissions 26
example
denying 27
granting 29
adding logic 70
application administration 52, 55
application files
form 14
protecting 15
relationships 15
viewing 11
application scope
configure layout 47
creating lists and forms 44, 44
form design 44, 46
namespace identifier 8
creation options
create a custom application 62
start from a template 62
start from an existing service 62, 63
start from global 63
start from scratch 61
custom application record 6
data tables 64
default view 64
deleting 87
deploy 55
developers 71
development environment 239
fix scripts
running 19
testing 18
include data 83
install 77
install updates 78
managing 76
moving 89
parts of 4
publish 85
publishing 82, 324
related lists 64
removing 87
repository 82, 324
selecting 78, 78
sharing 81
studio 243
uninstall 88
update set 85
updating 250
User interface 64
Applications module 4
approval ui macro 1976
approve execution plan
use script 1895
arbitrary JavaScript 2026
ArrayUtil
concat(Array parent, Array child) 400
contains(Array array, Object element) 401
convertArray(Object a) 401
diff(Array a, Array b) 402
ensureArray(Object object) 402
indexOf(Array array, Object element, Number
startIndex) 403
indexOf(Array array, Object element) 403
intersect(Array a, Array b) 404
union(Array a, Array b) 404
unique(Array a) 405
ArrayUtil API 400
AssessmentCreation
createAssessments() 406
AssessmentCreation API 405
Attachment API
DELETE 1604
GET 1606
GET file 1612
GET meta 1609
POST 1613
POST multipart 1617
role requirements 2092
authentication portal
create
asp.net 1991
autofill 2003
automated test administration 118
automated test framework 109, 110, 111, 111, 111, 112,
116, 116, 116, 117, 117, 117, 117, 118, 118, 119, 121,
122, 122, 122, 123, 123, 124, 124, 124, 128, 128, 131, 131,
135, 173, 214, 228, 235, 236
Automated test framework 108
Automated Test Framework
Add Item to Shopping Cart 188
admin properties 225
administering REST 126
assert json response payload element 203, 204
assert JSON response payload element 205
assert response header 201
assert response JSON payload 202
assert response time 201
assert response XML payload 202
assert status code 199
assert status code name 200
assert value 177

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
auto flush 237  
basic authentication profiles 126  
category 238  
clip modal 180  
execute server script 194  
field validation 178  
filter rest headers 127  
Open a Catalog Item 183  
Open a Record Producer 183  
open existing form 176  
open new form 175  
Order Catalog Item 189  
Open a Record Producer 189  
record delete 193  
record insert 192  
record query 191  
record update 192  
record validation 194  
replay request item 195  
REST 197  
REST API Explorer 197  
REST properties 127  
retention policy 237  
run module 221  
send REST 197  
send REST Request - Inbound 198  
Set Catalog Item Quantity 185  
send REST Request 197  
Set Variable Values 185  
set value 176  
Set Variable Values 185  
step categories 238  
step config 227  
step config types 175  
step environments 237  
Submit a Catalog Item 189  
table cleanup 237, 237  
test administration 225  
test client runner 221, 221  
test module 207  
test queue 222, 224, 225  
test record form 207  
test results 208, 208, 210  
test step config 227, 238  
test suite module 207  
test suite queue 225  
test suite record 207  
test suite results 212, 212, 219, 219, 220  
test template 238  
test templates 238  
UI visibility 179  
Validate Price and Recurring Price 187  
Validate Variable Values 186  
Variable State Validation 186  
Automated Test Framework> 195  
Automated Test log step> 195  
automated test step order 116, 228  
automated test suite 109, 111, 111, 112  
AutomationException 408  

B  
back reference 108, 135  
background scripts 1878  

baseline version 315  
breakpoints  
remove 2021  
set 2021  
Build tests verify functionality  
Scripted REST APIs 2113  
business 1849  
Business rule 138  
Business Rule and Workflow order 1817  
Business Rule order 1817  
business rules  
creating 1853  
how they work 1849  
scoped applications 1852  

C  
c sharp integration 2259  
Cabrillo  

isNativeO 408  
cabrillo button styles 409  
cabrillo close button styles 409  
cabrillo constants 409  
cabrillo data types 409  
cabrillo modal presentation styles 409  
cabrillo.attachment 410  
cabrillo.attachments  
addFile(String tableName, String sysId, Object params, String options) 413  
viewFile(Cabrillo.Attachment attachment, Cabrillo.Rect sourceRect, String sourceBase64Image) 414  
cabrillo.button 410  
cabrillo.camera  
getBarcode() 416  
cabrillo.geolocation  
getCurrentLocation() 416  
cabrillo.list  
selectItem(String title, String tableName, String query, Cabrillo.ListSelection selectedItem, Object params) 417  
selectItems(String title, String tableName, String query, Array selectedItems, Object params) 418  
cabrillo.listselection 411  
cabrillo.location  
cabrillo.locationcoordinate 412  
cabrillo.modal  
dismissModal(Object results) 420  
presentModal(String title, String url, String closeButtonStyle, String modalPresentationStyle) 420  
cabrillo.modalresponse 412  
cabrillo.navigation  
goBack() 421  
goto(String url, Cabrillo.NavigationRequest request) 422  
cabrillo.navigationrequest 412  
cabrillo.rect 413  
cabrillo.viewLayout  
hideSpinner() 423  
setBottomButtons(Array buttons, Function handler) 424  

© 2020 ServiceNow, Inc. All rights reserved.  
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
setNavigationBarButtons( Array buttons, Function handler) 425
setTitle( String title) 427
showSpinner() 428
cancel automated test 117, 117
cancel queued 117
cancel queued automated test suite 117
cancel test suite 117
CartJS
canViewRF() 461
cartDetails() 462
Catalog 1973
CatalogItemVariable
create(Boolean standardUpdate) 463
deleteRecord(Boolean standardUpdate) 464
read(Object columns, Boolean standardUpdate) 464
setAttributes(Object attributes) 464
update(Object columnValues, Boolean standardUpdate) 465
CatalogItemVariableSet
create(Boolean standardUpdate) 471
deleteRecord(Boolean standardUpdate) 471
read(Object columns, Boolean standardUpdate) 471
setAttributes(Object attributes) 472
update(Object columnValues, Boolean standardUpdate) 472
CatalogItemVariableSetM2M
create(Boolean standardUpdate) 473
deleteRecord(Boolean standardUpdate) 473
read(Object columns, Boolean standardUpdate) 473
setAttributes(Object attributes) 474
update(Object columnValues, Boolean standardUpdate) 474
CatalogJS
canView(boolean mobile) 429
canView(String gr) 428
Catalog(Glide Record gr) 428
canView(String sys_id) 428
getAvailableCatalog() 429
getBackgroundColor() 430
getCatalogCount() 430
getCategories() 431
getCatalogIds() 432
getDescription() 433
getDesktopImageSRC() 433
getGr() 434
getHeaderIconSRC() 436
getID() 437
getTitle() 437
hasCategories() 438
hasItems() 438
isWishlistEnabled() 439
CatCategory
canView(Boolean isMobile, String userSysId) 476
create(Boolean standardUpdate) 476
deleteRecord(Boolean standardUpdate) 477
getID() 479
read(Object columns, Boolean standardUpdate) 477
setAttributes(Object attributes) 478
setTableName(String tableName) 478
update(Object columnValues, Boolean standardUpdate) 479
Category 2249, 2251
Cattitim
API 439
canViewOnSearch(boolean isMobile) 439
create(Boolean standardUpdate) 443
deleteRecord(Boolean standardUpdate) 443
getFirstAccessibleCategoryForSearch(String catalogId) 440
getRecordClass() 441
isVisibleServicePortal() 441
setAttributes(Object attributes) 444
setCatalogs(String catalogs) 444
setCategories(String categories) 444
setImage(String dbImageSysId, String type) 445
setTableName(String tableName) 445
update(Object columnValues, Boolean standardUpdate) 445
CertificateEncryption
CertificateEncryption() 480
generateMac(key, algorithm, data) 480
getThumbPrint(String certificateID, String algorithm) 481
getThumbPrintFromKeystore(String certificateID, String alias, String algorithm) 481
sign(String certificateID, String alias, String aliaspassword, String algorithm, String datatosign) 482
certificates
cryptography
APIs 604, 604
ChangeCollisionHelper
addCitToChgAffectedCis(String ci, String changeld) 484
getAffectedCisByChangeld(String changeld) 485
getBlackoutsByDate(GlideDateTime startDate, GlideDateTime endDate) 485
getChangesWithAffectedCi(String ci, GlideDateTime startDate, GlideDateTime endDate) 485
getChangesWithCi(String ci, GlideDateTime startDate, GlideDateTime endDate) 486
calciMaintenanceSchedule(String ci) 486
calciDependencies(String ci) 487
isCalciAffectedCis(String ci, String changeld) 487
isDateInCalciMaintenanceWindows(GlideDateTime startDate, GlideDateTime endDate, String maintenanceWindow) 487
ChangeCollisionHelper API 484
ChangeConflict
ChangeConflict(String ciId, String changeld, String type) 488
toString() 488
ChangeConflict API 488
ChangeConflictHandler
addChangeConflict(String chgConflict) 489
ChangeConflictHandler() 489
deleteConflictsByChangeld(String changeld) 489
getConflicts() 489
saveConflicts() 490
ChangeConflictHandler API 489
child test suite 111
CI Lifecycle Management API
DELETE 1620, 1622
GET 1623, 1624, 1625, 1626, 1627, 1628, 1629
PATCH 1631
POST 1632, 1633, 1634
CI Lifecycle Management API 1620
CIData
addRelatedList(Array relatedList) 491
CIData(String debugFlag) 491
convertRelatedList(String sensor, String tableName, String refField, String keyName) 491
fromXML(String xml) 491
dataGet() 492
getM2MTable(String table, String refField) 492
getRelatedList(String table, String refField) 493
getRelatedListInstance(String table, String refField) 493
init() 494
loadFromCI(String cmdb_ci) 494
toString() 495
toXML() 495
CIData API 490
CIIdentification
CIIdentification(String ciData, String logger) 495
dbGet(String msg) 496
process() 498
CIIdentification API 495
CIIdentifierResult
CIIdentifierResult(Array matched, Boolean matchable, Boolean terminatedChain) 496
CIIdentifierResult API 496
CimCIData
addSerial(String type, String serial) 497
SerialRecord(String type, String serial) 497
setMakeAndModel(String make, String model) 498
CimCIData API 497
CimIDSensor
configureTriggredProbe(String probe) 498
getInstanceHashToken() 498
prepare() 499
process(Object results, CimCIData cimData, CimIDSensor sensor) 499
runMultiProbeScript(String script, Object probeResult, String probeRecord) 499
updateDeviceCount() 500
CimIDSensor API 498
CimInstanceToken
getHashToken(String cimomip) 500
parse(XMLObj instance) 501
CimInstanceToken API 500
CimProbe
getNamedQueries() 501
getQueries() 501
getQueryRecords() 502
CimProbe API 501
CimProbeResult
getNamedInstances(String namedQueries) 502
getQueries() 502
CimProbeResult API 502
CIRelatedList
addRec(String record) 503
addRecs(String records) 503
appendXMLChildFld(String parent, String name, Stringdatum) 503
appendXMLChildRecord(String parent, String name, String record) 504
cHECKSysCollection() 504
cHECKSysM2M() 504
CIRelatedList(String table, String refField, String cmdb_ci, String debugFlag) 505
fromXML(String element) 505
populate() 505
remove() 506
toString(Array result) 506
toXML(String document, String element) 506
xmlToRecord(String records, String element) 507
CIRelatedList API 503
CIUtils
servicesAffectedByCI(String CI_sys_id) 482
servicesAffectedByTask(GlideRecord task) 483
CIUtils API 482
ClassifierProbes
launch() 507
launchSupplementary() 507
size() 508
ClassifierProbes API 507
ClassName
methodName() 1286, 1288
click UI action 181
Client
call(Object identifier) 1293
Client(Object) 1294
forwardCall(Object argument) 1294
hangupCall() 1295
mute(Boolean muted) 1295
Notify WebRTC Client 1293
pickupCall() 1296
client script
best practices 1904
code, enclose 1908
DOM manipulation 1908
global 1908
client scripts
service catalog 1913
Service Catalog 1913, 1913, 1914
UI scripts 1903, 1903, 1904
Client Scripts
Service Catalog 1912
Client side GlideRecord
GlideRecord(String tableName) 905
Clientside GlideRecord
glAdOrderBy(String column) 900
addQuery(String name, Object operator, Object value) 902
addQuery(String name, Object value) 901
addQuery(String query) 900
deleteRecord(Function responseFunction) 903
get(Object sys_id) 904
getEncodedQuery() 904
getLimit() 904
getTableNames() 905
hasNext() 905
insert(Function responseFunction) 906
next() 906
orderBy(String column) 907
query(Function responseFunction) 908
setLimit(Number maxQuery) 908

Clientside GlideRecord API 900
Clientside GlideRecordV3
addOrderBy(String column) 993
addQuery(String fieldName, Object operator, Object value) 994
addQuery(String fieldName, Object value) 993
deleteRecord(Function responseFunction) 994
get(String sysId) 994
getEncodedQuery() 995
getTableName() 995
GlideRecord(String tableName) 995
hasNext() 996
insert(Function responseFunction) 996
next() 996
orderBy(String column) 997
query() 997

CMDB instance API
DELETE 1647
GET 1648, 1649
PATCH 1652
POST 1653, 1655
PUT 1656

CMDB meta API
GET 1658

CMDBGroup API
getAllCIs(String groupId, Boolean requireCompleteSet) 509
getAllCIsFromQueryBuilder(String groupId, Boolean requireCompleteSet) 511
getManualCIList(String groupId, Boolean requireCompleteSet) 513
getSavedQueryIdList(String groupId, Boolean requireCompleteSet) 515
setManualCIList(String groupId, String ciSysIds) 517
setSavedQueryIdList(String groupId, String queryIds) 519

CMDBTransformUtil
CMDBTransformUtil() 521
generateError() 521
geriveOutputPayload() 522
geriveOutputRecordSysId() 524
hasError() 525
identifyAndReconcile (Object source, Object map, Object log) 525
logTransformStats(Object log) 526

CMDBUtil
setDataSource(String source) 527

ConnectActionResponse
newRecord(String table, Object values, String view) 533

ConnectionInfo
getAttribute(name) 535
geriveCredentialAttribute(String name) 535
geriveDataMap() 536
geriveExtendedAttributes() 536

ConnectionInfoProvider
getConnectionInfo() 538
geriveConnectionInfoByDomain(String aliasID, String domainID) 538

Conversation
create(String name) 539
sendMessage(String body, String field) 541

CreateNow Studio
keyboard shortcuts 244

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
CriteriaEvaluator
    addCriterion(String lhComparand, String operator, String rhComparand) 542
    evaluate(Object lhValues, String rhValues, Boolean any) 542
CriteriaEvaluator API 542
cross-scope access 37, 38, 39
CSV web service
    file, posting 2265
    parameters 2265
custom 1976
custom application development 4
custom category 121
Custom Chart Data Generation 1478
custom processor
    create 1877
custom scripted step config 196
custom step 118
custom step config 119
CustomEvent
    fireEvent(String event, String qualifier) 543
customizations
    load from XML 353

data policy 144
Date
    validate 2007
DateTimeUtils
    int8ToGlideDateTime(Number int64) 549
    msToGlideDateTime(Number milliseconds) 549
DateTimeUtils API 548
date 2016, 2026
date 2016
    debug 2016, 2026
date 2016
    debug script 2042
date 2016
    debugger
        User interface 2016
date 2016
    debugging
        session 2026
date 2016
    UI policies 2038
date 2016
    debugging classifications 2032
date 2016
    delegated development 75
date 2016
    dependencies 10
date 2016
    design access 24, 69
date 2016
    developer permissions 75
date 2016
    developers
        group 73
        user 73
date 2016
    development process 58
DHTMLX 1330
Discovery
    script includes
        using 1961, 1962
DSN
    global default 2344
Duration Field Value
    setting 1821
DurationCalculator
    DurationCalculator() 553
    relative duration 1883
    simple duration 1881
DurationCalendar
    calcDuration() 550
    calcRelativeDueDate() 550
    calcRelativeDuration() 551
    calcScheduleDuration() 552, 552
    getEndDateTime() 553
    getSeconds() 553
    getTotalSeconds() 554
    isAfter() 554
    setSchedule() 555
    setStartDateTime() 555, 556
    setTimeZone() 556
E
    edit test template 122
    elements
        adding
            scrolling 2284
            scrolling
                adding 2284
e-mail 1247
Email API
    GET 1660
    POST 1665
enable tests 123
Enforce and test access controls
    Scripted REST APIs 2113
enforce subscriptions 57
ES5 standards mode 2015
ES5 Standards Mode 2015
escaping types
    jelly 1928
example
    incident
        priority 1 2284
        priority 1
            incident 2284
Examples
    end to end tutorial 2255, 2256, 2256, 2258, 2259, 2259, 2260, 2261
    web services c sharp .net 2255, 2256, 2256, 2258, 2259, 2259, 2260, 2261
EXCEL
    web service 2263
Excel web service
    parameters 2264
    URL parameters 2263
execution plan
    script 1891
    execution plan scripts
        limitations 1892
        write 1892
ExpenseAllocation
    createAllocation() 557
    initialize() 558
ExpenseLine
    () 559
    createExpense() 558
    methodName() 560
    processCIParents() 559
    setCostSource() 560
fetch XML
  use code 2263
field messages
display 1981
field types
  scripting 1819
field value
  null 1819
field values
  business rules and client scripts for controlling 1818
Field Watcher 2033, 2033, 2034, 2035, 2035, 2036
Follow REST API conventions
  Scripted REST APIs 2111
FormInfoHeader
    addMessage(String message) 562
free-form script 2026
freeform script 2026
fulfillment tables 19, 20, 57
Fullcalendar 1330
function 1863
G
  GlideAggregate 1941, 1941
GlideAggregate
    addAggregate(String agg, String name) 570
    addEncodedQuery(String query) 571
    addHaving(String name, String operator, String value) 572
    addTrend(String fieldName, String timeInterval) 573
    getAggregate(String agg, String name) 574
    getQuery() 574
    getTotal(String agg, String name) 575
    getValue(String name) 576
    groupBy(String name) 576
    orderBy(String name) 577
    orderByAggregate(String agg, String name) 578
    query() 579
    setGroup(Boolean b) 580
GlideAjax
    addParam() 594
    example asynchronous 1920
    example synchronous 1922
    getAnswer() 594
    getXML() 595
    getXMLAnswer(Function callback) 596
    getXMLWait() 596
    GlideAjax() 597
GlideAjaxV3
    addParam(String name, String value) 597
    getJSON(Function callback) 598
    getParam(String name) 598
    getParams() 598
    getProcessor() 599
getURL() 599
getXML(Function callback) 599
getXMLAnswer(Function callback) 600
GlideAjax(String processor, String targetURL) 601
GlideAjax(String processor) 600
setErrorCallback(Function callback) 601
setProcessor(String serverScript) 601
getXML() 599
getXMLAnswer() 600
GlideAjaxV3 API 597
GlideappCalculationHelper API 602
GlideCertificateEncryption
generateMac(key, algorithm, data) 603
GlideCertificateEncryption() 605
sign(certificateID, alias, aliaspassword,algorithm,datatosign) 605
GlideConversation
description 606
document 606
name 607
queueEntry 607
sys_id 607
table 608
type 608
GlideConversation API 606
GlideDate API 1947
GlideDateTime
add(GlideTime time) 614
add(Number milliseconds) 614
addDays(Number days) 615
addDaysLocalTime(Number days) 615
addDaysUTC(Number days) 616
addMonths(Number months) 619
addMonthsLocalTime(Number months) 620
addMonthsUTC(Number months) 620
addSeconds(Number seconds) 617
addWeeks(Number weeks) 617
addWeeksLocalTime(Number weeks) 618
addWeeksUTC(Number weeks) 618
addYears(Number years) 621
addYearsLocalTime(Number years) 621
addYearsUTC(Number years) 622
compareTo(GlideDateTime dateTime) 623
equals(Object GDT) 623
gDATE() 624
gDayOfMonth() 625
gDayOfMonthLocalTime() 625
gDayOfMonthUTC() 626
gDayOfWeek() 626
gDayOfWeekLocalTime() 627
gDayOfWeekUTC() 627
gDaysInMonth() 628
gDaysInMonthLocalTime() 628
gDaysInMonthUTC() 628
gDisplayValue() 630
gDisplayValueInternal() 630
gDSTOffset() 631
gError() 632
gInternalFormattedLocalTime() 632
gInternalMidnight() 633
gLocalDate() 633
gLocalTime() 634
gMonth() 634
getMonthLocalTime() 635
getMonthUTC() 635
getNumericValue() 636
getSpanTime(Number dayOfWeek) 636
gTime() 637
gTZOffset() 637
getUserFormattedLocalTime() 638
gUserTimeZone() 638
getUTCDate() 639
getWeekOfYearLocalTime() 639
getWeekOfYearUTC() 639
gValue() 640
getWeekOfYearLocalTime() 640
getWeekOfYearUTC() 641
getYear() 641
getYearLocalTime() 642
getYearUTC() 642
GlideDateTime() 643
GlideDateTime(GlideDateTime gDT) 644
GlideDateTime(String dateTime) 643
hasDate() 644
isDST() 645
isValid() 645
setDayOfMonth(Number day) 646
setDayOfMonthLocalTime(Number day) 646
setDayOfMonthUTC(Number day) 647
setDisplayValue(String asDisplayed) 648
setDisplayValue(String dateTime, String format) 648
setDisplayValueInternal(String dateTime) 649
setDisplayValueInternalWithAlternates(String dateTime) 650
setGlideDateTime(GlideDateTime gDT) 650
setInitialValue(String dateTime) 651
setMonth(Number month) 651
setMonthLocalTime(Number month) 652
setMonthUTC(Number month) 652
setNumericValue(Number milliseconds) 653
setTZ(TimeZone timeZone) 653
setUTCMidnight(Number day) 654
setValue(Object dateTime) 654
setValueUTC(String dateTime, String format) 654
setYear(Number year) 657
setYearLocalTime(Number year) 657
setYearUTC(Number year) 658
subtract(GlideDateTime start, GlideDateTime end) 659
subtract(GlideTime time) 658
subtract(Number milliseconds) 659
toString() 660
GlideDateTime API 1947
GlideDialogWindow
adjustBodySize() 698
destroy() 699
GlideDialogWindow() 699
render() 701
setPreference() 701
setSize() 702
setTitle() 702
GlideDocumentV3
elemelement(String selector, Element context) 709
elemelements(String selector, Element context) 709
GlideDocumentV3 API 708
GlideElement
canCreate() 715, 740
canRead() 715, 740

canWrite() 716, 740
changes() 716
changesFrom(Object value) 717
changesTo(Object value) 718
dateNumericValue() 719
debug(Object o) 719
ggetAttribute(String attributeName) 720, 743
gBaseTableName() 720
ggetChoices(String value) 721
ggetChoiceValue() 722
ggetDebugCount() 722
ggetDependent() 722
ggetDependentTable() 723
ggetDisplayValue(Number maxChar) 723
ggetDisplayValueExt(Number maxChar, String nullSub) 724
ggetED() 724
ggetElementValue(String value) 725
ggetError() 725
ggetEscapedValue() 725
ggetFieldStyle() 726
ggetGlideObject() 726
ggetGlideRecord() 727
ggetGlideEncrypter().decrypt(String encryptedString) 765
getGlideEncrypter().encrypt(String clearString) 765
GlideEncrypter() 766
GlideEncrypter(String key) 766
GlideExcelParser().close() 772
ggetColumnHeaders() 773
ggetErrorMessage() 773
ggetRow() 773
GlideExcelParser().next() 774
parse(GlideScriptableInputStream inputStream) 775
setHeaderRowNumber(Number headerRowNumber) 775
setNullToEmpty(Boolean empty) 775
setSheetName(String sheetName) 776
setSheetNumber(Number sheetNumber) 776
GlideEncrypter()
addDecoration(String fieldName, String icon, String title, String color) 779
addDecoration(String fieldName, String icon, String title) 778
addErrorMessage(String message) 780
addInfoMessage(String message) 781
addOption(String fieldName, String choiceValue, String choiceLabel, Number choiceIndex) 782
addOption(String fieldName, String choiceValue, String choiceLabel) 781
clearMessages() 782
clearOptions(String fieldName) 783
clearValue(String fieldName) 783
disableAttachments() 783
enableAttachments() 784
flash(String widgetName, String color, Number count) 784
ggetName() 785
ggetBooleanValue(String fieldName) 785
ggetControl(String fieldName) 785
ggetDecimalValue(String fieldName) 786
ggetElement(String id) 786
getFormElement() 787
ggetHelpTextControl(String fieldName) 787
getIntValue(String fieldName) 787
ggetLabelOf(String fieldName) 788
getOption(String fieldName, String choiceValue) 788
ggetOption(String fieldName, String choiceValue) 788
ggetReference(String fieldName, Function callBack) 789
ggetSectionNames() 790
ggetSections() 790
ggetTableName() 791
ggetUniqueValue() 791
ggetValue(String fieldName) 791
hideAllFieldMsgs() 792
hideAllFieldMsgs(String type) 792
hideErrorBox(String input) 792
hideFieldMsg(String fieldName, Boolean clearAll) 793
hideFieldMsg(String fieldName) 793
hideRelatedList(String listTableName) 794
hideRelatedLists() 794
isLiveUpdating() 794
isMandatory(String fieldName) 795
isNewRecord() 795
isSectionVisible(String sectionName) 796
refreshSlushbucket(String fieldName) 796
removeDecoration(String fieldName, String icon, String title, String color) 797
removeDecoration(String fieldName, String icon, String title) 797
removeOption(String fieldName, String choiceValue) 796
save() 798
setDisabled(String fieldName, Boolean disable) 798
setDisplay(String fieldName, Boolean display) 799
setLabelOf(String fieldName, String label) 799
setMandatory(String fieldName, Boolean mandatory) 800
setReadOnly(String fieldName, Boolean readOnly) 800
setSectionDisplay(String sectionName, Boolean display) 802
setValue(String fieldName, String value) 801
setValue(String fieldName, String value, String displayValue) 801
setVisible(String fieldName, Boolean display) 803
showErrorBox(String input, String message) 806
showErrorBox(String name, String message, Boolean scrollForm) 804
showFieldMsg(String field, String message, String type) 804
showFieldMsg(String field, String message, String type, Boolean scrollForm) 805
showRelatedList(String listTableName) 806
showRelatedLists() 806
submit() 807
submit(String verb) 807
GlideGuidV3 generate(Number stringLength) 811
GlideGuidV3 API 811
GlideImpersonate canDebug() 811
canImpersonate() 812
impersonate() 813
isImpersonating() 813
GlideList2 addFilter() 814
get() 814, 815
getchecked() 815
getfixedQuery() 816
getGroupBy() 816
getListName() 817
getOrderBy() 817
getParentTable() 818
getQuery() 818
getRelated() 819
getTableName() 819
getTitle() 820
getView() 820
isUserList() 821
refresh() 821
refreshWithOrderBy() 822
setFilter() 823
setFilterAndRefresh() 823
setFirstRow() 824
setGroupBy() 824
setOrderBy() 825
setRowsPerPage() 826
showHideGroups() 826
showHideList() 827
sort() 827
sortDescending() 828
toggleList() 828
toggleListNoPref() 829
GlideListProperties getListID() 843
getTitle() 843
GlideListProperties getListName() 842
hasActions() 844
hasBottomNav() 844
hasBottomVCR() 845
hasFilter() 845
hasHeader() 846
hasHeaderContextMenu() 846
hasListMechanic() 847
hasPopup() 847
hasRowContextMenu() 848
hasSearch() 848
hasTitle() 849
hasTitleContextMenu() 849
hasTopVCR() 850
isOmitFilter() 850
isSaveFilterHidden() 851
isShowLinks() 851
isToggleHeader() 852
setCanChangeView() 852
setCanGroup() 853
setCanSort() 853
setContextMenus() 854
setHasActions() 854
setHasBottomNav() 855
setHasBottomVCR() 856
setHasBreadcrumbs() 855
setHasFilter() 856
setHasHeader() 857
setHasHeaderContextMenu() 857
setHasListMechanic() 858
setHasPopup() 858
setHasRowContextMenu() 858
setHasSearch() 859
setHasTitle() 859
setHasTitleContextMenu() 860
setHasTopVCR() 860
setHideRows() 861
setListID(String sys_ID) 862
setListName(String name) 862
setSaveFilterHidden() 861
setShowLinks() 862
setTitle(String title) 863
setVCR() 864
GlideListV3 addFilter(String filter) 829
getObject(DomElement) 830
getString(listId) 830
getchecked() 830
getfixedQuery() 831
getFormTarget() 831
getGroupBy() 831
getListName() 832
getOrderBy() 832
getParentTable() 832
getQuery(Object options) 833
getReferringUrl() 833
getRelated() 834
getRelatedListType() 834
getRelationshipId() 834
getRowCount() 835
getRowPerPage() 835
getTableName() 835
getView() 836
isUserList() 836
refresh(Number firstRow, Object additionalParams) 836
refreshWithOrderBy(Number firstRow, Object additionalParams) 837
setFilter(String filter, Boolean saveOrderBy, Boolean saveGroupBy) 837
setFilterAndRefresh(String filter) 838
setFirstRow(Number firstRow) 838
setFormTarget(String target) 838
setGroupBy(String groupBy) 839
setOrderBy(String orderBy) 839
setReferringUrl(String url) 840
setRowsPerPage(Number numRows) 840
showHideGroups(Boolean showFlag) 840
showHideList(Boolean showFlag) 841
sort(String field) 841
sortDescending(String field) 841
toggleList() 841
toggleListNoPref() 842
GlideListV3 API 829
GlideMenu
clearImage(GlideMenuItem item) 866
clearSelected() 866
getFilter(String itemID) 867
setDisabled(GlideMenuItem item) 867
setEnabled(GlideMenuItem item) 868
setHidden(GlideMenuItem item) 868
setImage(GlideMenuItem item, String imgSrc) 868
setLabel(GlideMenuItem item, String label) 869
setVisible(GlideMenuItem item) 869
GlideMenu API 866
GlideModalFormV3
addParm(String name, String value) 870
GlideModalForm(String title, String tableName, Function onCompletionCallback, Boolean readOnly) 870
render() 871
setCompletionCallback(Function callbackFunction) 871
setOnloadCallback(Function callbackFunction) 872
setSysId(String sys_id) 872
GlideModalFormV3 API 870
GlideModalV3
clearImage(GlideMenuItem item) 866
clearSelected() 866
getFilter(String itemID) 867
GlideModalV3 API 873
GlideNavigationV3
open(String url, String target) 878
openPopup(String url, String name, String features, Boolean noStack) 878
openRecord(String tableName, String sys_id) 879
refreshNavigator() 879
reloadWindow() 880
GlideNavigationV3 API 878
GlideNotificationV3
show(String type, String message) 880
GlideNotificationV3 API 880
GlideOAuthClient
token(String oauthEntityName, String requestor) 881
requestToken(String clientName, String jsonId) 882
requestTokenByRequest(String clientName, GlideOAuthClientRequest request) 883
revokeToken(String clientName, String accessToken, String refreshToken, GlideOAuthClientRequest request) 883
GlideOAuthClientRequest
getGrantType() 884
getHeader(String name) 884
getPassword() 885
getRefreshToken() 885
GlideOAuthClientRequest API 884
GlideOAuthClientResponse	getAccessToken() 891
token() 891
getSysId(String sys_id) 892
GlideOAuthClientResponse API 891
GlideQueryCondition
addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord
_next() 943
_query() 946, 985
GlideRecord API 943
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
gliderecord 1884
GlideQueryCondition API 894
GlideRecord search 909
GlideOAuthClientResponse getAccessToken() 891
GlideOAuthClientResponse getSysId(String sys_id) 892
GlideQueryCondition addCondition(String name, String oper, Object value) 894
addOrCondition(String name, String oper, Object value) 895
addFunction(Object function) 911
addInactiveQuery() 912
addJoinQuery(String table, String primaryField, String joinTableField) 914
addJoinQuery(String table, String primaryField) 914
addJoinQuery(String table) 912
addNullQuery(String field, Name) 915
addNullQuery(String table) 916
addQuery(String name, Object operator, Object value) 917
applyTemplate(String template) 919
autoSysFields(Boolean e) 919
canCreate() 920
canDelete() 921
canRead() 921
canWrite() 922
changes() 922
deleteMultiple() 923
deleteRecord() 923
find(String columnName, String value) 924
goToRecord(Object name, Object value) 925
getClassDisplayValue() 926, 969
getDisplayValue() 927
gET() 927, 971
gElement(String fieldName) 928
genEncodedQuery() 929
genEscapedDisplayValue() 929
genFields() 929
getLabel() 920, 973
getLink(Boolean noStack) 931
getLocation(Boolean b) 932
gPlural() 932
gRelatedLists(Boolean b) 934
gRelatedTables(Boolean b) 934
goRowCount() 934
goRowNumber() 935
goTableName() 935
gValue(String fieldName) 936
GlideRecord(String tableName) 936
hasAttachments() 936
hasNext() 937
initialize() 938
insert() 938
insertWithReferences() 939
instanceOf(String className) 940
isValid() 940
isValidField(String fieldName) 941
isValidRecord() 941
newRecord() 942
next() 942
operation() 944
orderBy(String fieldName) 944
orderByDesc(String fieldName) 945
query tables 1884, 1885, 1887
query(Object field, Object value) 945
restoreLocation() 947
saveLocation() 948
setAbortAction(Boolean b) 948
setDisplayValue(String name, Object value) 949
setForceUpdate(Boolean force) 949
setLimit(Number limit) 949
setLocation(Number rowNumber) 950
setNewGuid() 950
setNewGuidValue(String guid) 951
setQueryReferences(Boolean queryReferences) 951
setUseEngines(Boolean e) 952
setWorkflow(Boolean e) 953
update(Object reason) 954
updateMultiple() 955
updateWithReferences(Object reason) 955
GlideRecord variable set to null 1898
GlideRecordSecure 1945
GlideRecordUtil getCGIR(String sys_id) 989
getFields(GlideRecord gr) 990
getGR(String baseTable, String sys_id) 990
getTables(String tableName) 991
mergeToGR(Object hashMap, GlideRecord gr, Object ignore) 991
populateFromGR(Object hashMap, GlideRecord gr, Object ignore) 992
GlideRecordUtil API 989
GlideScopedEvaluator evaluateGQL(String gQL, Object variables) 767, 770
getVariables(String name) 768, 771
GlideScopedEvaluator() 768, 771
putVariable(String name, Object value) 769
GlideServletRequest 1007
GlideServletResponse 1011
GlideSession clearClientData(String paramName) 1013
clientData(String paramName) 1013
gGetLanguage() 1014
gGetRoles() 1015
ggetTimeZoneName() 1015
isInteractive() 1016
isLoggedln() 1016
putVariable(String name, String value) 1017
GlideServletResponse 1011
GlideSession clearClientData(String paramName) 1013
clientData(String paramName) 1013
gGetLanguage() 1014
gGetRoles() 1015
isInteractive() 1016
isLoggedln() 1016
putVariable(String name, String value) 1017
GlideSession clearClientData(String paramName) 1013
clientData(String paramName) 1013
gGetLanguage() 1014
gGetRoles() 1015
isInteractive() 1016
isLoggedln() 1016
putVariable(String name, String value) 1017
GlideSPScriptable getRecordVariablesArray(GlideRecord gr) 1038
GlideSystem addErrorObject(Object message) 1051
addInfoMessage(Object message) 1052
addMessage(String type, Object message) 1052
beginningOfLastMonth() 1053, 1105
beginningOfLastWeek() 1053, 1105
beginningOfNextMonth() 1054, 1106
beginningOfNextWeek() 1054, 1106
beginningOfNextYear() 1055, 1106
beginningOfThisMonth() 1055, 1107
beginningOfThisQuarter() 1056, 1107
beginningOfThisWeek() 1056, 1107
beginningOfThisYear() 1056, 1108
beginningOfToday() 1057
beginningOfTomorrow() 1058
beginningOfYesterday() 1058
calDateDiff(String startDate, String endDate, Boolean numericValue) 1059

addFunction(Object function) 911
addJoinQuery(String table, String primaryField, String joinTableField) 914
addJoinQuery(String table, String primaryField) 914
addJoinQuery(String table) 912
addNullQuery(String fieldName) 915
addNullQuery(String table) 916
addQuery(String name, Object operator, Object value) 917
applyTemplate(String template) 919
autoSysFields(Boolean e) 919
canCreate() 920
canDelete() 921
canRead() 921
canWrite() 922
changes() 922
deleteMultiple() 923
deleteRecord() 923
find(String columnName, String value) 924
get(Object name, Object value) 925
getAttribute(String fieldName) 926, 969
getClassDisplayValue() 926, 970
gsetDisplayValue() 927
gET() 927, 971
gElement(String fieldName) 928
genEncodedQuery() 929
genEscapedDisplayValue() 929
genFields() 929
getLabel() 930, 973
gGetLink(Boolean noStack) 931
getLocation(Boolean b) 932
gPlural() 932
gRelatedLists(Boolean b) 934
gRelatedTables(Boolean b) 934
goRowCount() 934
goRowNumber() 935
goTableName() 935
getvalue(String fieldName) 936
GlideRecord(String tableName) 936
hasAttachments() 936
hasNext() 937
initialize() 938
insert() 938
insertWithReferences() 939
instanceOf(String className) 940
isValid() 940
isValidField(String fieldName) 941
isValidRecord() 941
newRecord() 942
next() 942
operation() 944
orderBy(String fieldName) 944
orderByDesc(String fieldName) 945
query tables 1884, 1885, 1887
query(Object field, Object value) 945
restoreLocation() 947
saveLocation() 948
setAbortAction(Boolean b) 948
setDisplayValue(String name, Object value) 949
setForceUpdate(Boolean force) 949
setLimit(Number limit) 949
setLocation(Number rowNumber) 950
setNewGuid() 950
setNewGuidValue(String guid) 951
setQueryReferences(Boolean queryReferences) 951
setUseEngines(Boolean e) 952
setWorkflow(Boolean e) 953
update(Object reason) 954
updateMultiple() 955
updateWithReferences(Object reason) 955
GlideRecord variable set to null 1898
GlideRecordSecure 1945
GlideRecordUtil getCGIR(String sys_id) 989
getFields(GlideRecord gr) 990
gGetGR(String baseTable, String sys_id) 990
getTables(String tableName) 991
mergeToGR(Object hashMap, GlideRecord gr, Object ignore) 991
populateFromGR(Object hashMap, GlideRecord gr, Object ignore) 992
GlideRecordUtil API 989
GlideScopedEvaluator evaluateGQL(String gQL, Object variables) 767, 770
getVariables(String name) 768, 771
GlideScopedEvaluator() 768, 771
putVariable(String name, Object value) 769
GlideServletRequest 1007
GlideServletResponse 1011
GlideSession clearClientData(String paramName) 1013
clientData(String paramName) 1013
gGetLanguage() 1014
gGetRoles() 1015
isInteractive() 1016
isLoggedln() 1016
putVariable(String name, String value) 1017
GlideServletResponse 1011
GlideSession clearClientData(String paramName) 1013
clientData(String paramName) 1013
gGetLanguage() 1014
gGetRoles() 1015
isInteractive() 1016
isLoggedln() 1016
putVariable(String name, String value) 1017
GlideSPScriptable getRecordVariablesArray(GlideRecord gr) 1038
GlideSystem addErrorObject(Object message) 1051
addInfoMessage(Object message) 1052
addMessage(String type, Object message) 1052
beginningOfLastMonth() 1053, 1105
beginningOfLastWeek() 1053, 1105
beginningOfNextMonth() 1054, 1106
beginningOfNextWeek() 1054, 1106
beginningOfNextYear() 1055, 1106
beginningOfThisMonth() 1055, 1107
beginningOfThisQuarter() 1056, 1107
beginningOfThisWeek() 1056, 1107
beginningOfThisYear() 1056, 1108
beginningOfToday() 1057
beginningOfTomorrow() 1058
beginningOfYesterday() 1058
calDateDiff(String startDate, String endDate, Boolean numericValue) 1059
dateDiff(String startDate, String endDate, Boolean numericValue) 1059
dateGenerate(String date, String range) 1060
daysAgo(Number days) 1061, 1108
daysAgoEnd(Number days) 1062, 1109
daysAgoLocal 1062
daysAgoStart(Number days) 1063, 1109
dayOfWeekLastMonth() 1063, 1111
dayOfWeekLastWeek() 1064, 1111
dayOfWeekLastYear() 1064, 1112
dayOfWeekNextMonth() 1064, 1112
dayOfWeekNextWeek() 1065
dayOfWeekNextYear() 1065, 1112
dayOfWeekThisMonth() 1066, 1113
dayOfWeekThisQuarter() 1066, 1113
dayOfWeekThisWeek() 1067, 1113
dayOfWeekThisYear() 1067, 1114
dayOfWeekToday() 1067
dayOfWeekTomorrow() 1068
dayOfWeekYesterday() 1068
eventQueue(String name, Object glideRecord, String parm1, String parm2, String queue) 1068
eventQueueScheduled(String name, Object glideRecord, String parm1, String parm2, Object expiration) 1069
flushMessages() 1070
getAvatar() 1070
gGetCurrentScopeName() 1071
gDisplayColumn(String tableName) 1071
gDisplayValueFor(String tableName, String recordID, String fieldName) 1072
gGetErrorMessages() 1072, 1119
gGetEscapedProperty(String key, Object substituteObject) 1073
gGetImpersonatingUserDisplayName() 1073
gGetImpersonatingUserName() 1073
gGetInfoMessages() 1074
gGetInitials() 1074
gMessage(String messageID, Object args) 1075
gMessage5(String messageID, Object args) 1075
gMessages(String message type) 1076
gNodeName(Object obj, Number index) 1077
gNodeValue(Object obj, Number index) 1077
gGetPreference(String key, Object default) 1077
gGetProperty(String key, Object alt) 1078
gGetScriptError(String script) 1078
gGetSession() 1079
gGetSessionId() 1079
gGetStyle(String tableName, String fieldName, String fieldValue) 1080
gGetTriviaMessages() 1080
gGetUser() 1080
gGetUserDisplayName() 1081
gGetUserId() 1081
gGetUserName() 1082
gGetUserNameByUserId(String sys_id) 1083
gXMLNodeList(String xml) 1083
gXMLText(String xml, String xpathQuery) 1083
hasRole(String roleName) 1084
hasRoleInGroup(Object roleName, Object group) 1084
hoursAgo(Number hours) 1085, 1125
hoursAgoEnd(Number hours) 1086, 1126
hoursAgoStart(Number hours) 1086, 1126
isFirstDayOfMonth(Object date) 1087
isFirstDayOfWeek(Object date) 1087
isFirstDayOfYear(Object date) 1087
isInteractive() 1088
isLastDayOfMonth(Object date) 1088
isLastDayOfWeek(Object date) 1089
isLastDayOfYear(Object date) 1089
isLoggedIN() 1089
isMobile() 1090
lastWeek() 1090
log(String message, String source) 1091
logError(String message, String source) 1091
logWarning(String message, String source) 1091
minutesAgo(Number minutes) 1092
minutesAgoEnd(Number minutes) 1093, 1130
minutesAgoStart(Number minutes) 1093, 1131
monthsAgo(Number months) 1094, 1130
monthsAgoEnd(Number months) 1094
monthsAgoStart(Number months) 1094, 1131
now() 1095
nowDateTime() 1095
nowGlideDateTime 1096
nowNoTZ() 1096
nowOffset() 1096
nowOffsetTime() 1096
quarterAgo(Number quarters) 1098
quartersAgoEnd(Number quarters) 1098, 1132
quartersAgoStart(Number quarters) 1099, 1132
setProperty(String key, String value, String description) 1099
setRedirect(Object URI) 1100
tableView(Object URI) 1101
tableExists(String tableName) 1101
userId() 1101
workflowFlush(Object glideRecord) 1102
yearsAgo(Number years) 1102
yesterday() 1103, 1136

GlideTimeline

groupByParent() 1149
registerEvent() 1150
setAutoRefresh() 1151
setDefaultPointIconClass() 1152
setExtraAjaxParam() 1152
setInitialViewRange() 1153, 1153, 1154, 1155
setReadOnly() 1155
showDependencyLines() 1156
showGridLines() 1156
showLeftPane() 1157
showLeftPaneAsTree() 1157
showLeftPaneInputBox() 1157
showLeftPaneText() 1157
showList() 1157
showTimelineText() 1157
showSummaryPane() 1158
showTimelineText() 1159
snapVertScrollingIntoRows() 1159
sortByDependencyLines() 1160
sortByLeftLabelText() 1160
sortByStartDate() 1160
sortByTimelineLabelText() 1160
snapshotPartial() 1160
贸易战() 1160
showTimelineLabelText() 1161
GlideTimelineItem

createTimelineSpan(String tableName, String sys_id) 1162
createTimelineSpan(String tableName) 1161
gGetImage() 1162
getIsDroppable() 1163
getLeftLabelText() 1163
getParent() 1163
getTimelineSpans() 1164
GlideTimelineItem(String tableName, String sys_id) 1164
GlideTimelineItem(String tableName) 1164
GlideURI get(String name) 1168
GlideURI get(String name, String value) 1169
toString(String path) 1170
GlideURLV3 addParam(String name, String value) 1170
GlideURLV3 API 1170
GlideUser
firstName 1172
getCompanyID() 1183
getAddress() 1184
getDomainDisplayValue() 1184
getDomainID() 1178
getEmail() 1185
getFirstName() 1185
getFullName() 1173
getID() 1186
getLastName() 1186
getMyGroups() 1187
getName() 1187
getRoles() 1188
getUserByID(String id) 1188
getUserRoles() 1189
hasRole() 1190
hasRole(String role, Boolean includeDefaults) 1173
hasRoleExactly(String role, Boolean includeDefaults) 1174
hasRoleFromList(String roles, Boolean includeDefaults) 1175
hasRoles(Boolean includeDefaults) 1176
isMemberOf() 1190
lastName 1176
userID 1176
userName 1177
global scope 7
global UI scripts 1903
grant access  
Update Set picker 334
GSLog 1982

Hierarchical test suites 214
Hierarchical tests 214

Hierarchical data
insert 2182

HistoryWalker
getUpdateNumber() 1193
getWalkedRecord() 1193
getWalkedRecordCopy() 1194
HistoryWalker(String tableName, String sysId, Boolean useAudit) 1192
HistoryWalker(String tableName, String sysId) 1192
isFieldLevelSecurity() 1195
isRecordLevelSecurity() 1196
isWithChanges() 1196
isWithJournalFields() 1197
isWithVariables() 1198
setFieldLevelSecurity(boolean fieldLevelSecurity) 1198
setRecordLevelSecurity(Boolean recordLevelSecurity) 1199
setWithChanges(Boolean withChanges) 1199
setWithJournalFields(Boolean withJournalFields) 1200
setWithVariables(Boolean withVariables) 1201
walkBackward() 1202
walkForward() 1202
walkTo(int updateCount) 1203

HostnameJS
format(String hostname, String source) 1204
getDomainName() 1204
getSysName() 1205
HostnameJS API 1204
HTTP client connection management 2331
HTTP connection management properties 2331
HTTP Response Codes 2097

i18NV3
format(String message, Object map) 1206
getMessage(String msgKey, Function callback) 1205
getMessages(Array msgKeys, Function callback) 1206

i18NV3 API 1205
ICalUtil 1209
ICalUtilSNC 1209

IdentificationEngine
createOrUpdateCI(String source, String input) 1210
identifyCI(String jsonString) 1214
runIdentificationAudit(GlideRecord gr) 1217
IdentificationEngineScriptableApi
createOrUpdateCI(String source, String input) 1217
identifyCI(String jsonString) 1222
runIdentificationAudit(GlideRecord gr) 1224
IdentifyReconcile API
   POST 1670, 1673
IDResult
   IDResult(Boolean explore, String sysID) 1224
IDResult API 1224
   immediately invoked function expression 1863
impersonate 190
Import Set API
   GET 1676
   POST 1678
in-bound access 37
inbound REST 173
Inbound REST
   basic auth 2091
      Multi-factor authentication 2096
      Multi-factor authentication responses 2096
   OAuth 2094
      OAuth example 2095
      required roles 2091
      response codes 2097
   security 2091
   supported headers 2096
   URI version 2091
inbound SOAP
   insert related records 2182
Inbound web services
   examples 2238
incident
   example
      priority 1 2284
      priority 1 example 2284
   installation settings 1878
integration
   source control 254
IPAddressFixup
      dedupe(String tableName, String ip) 1225
      dedupeAll() 1225
      fix() 1225
      getParentIP() 1226
      syncIP() 1226
IPAddressFixup API 1225
IPService
   creates 1227
   description 1227
   getFromArrayList(ArrayList) 1227
   IPService(Object source) 1227
   name 1228
   port 1228
   protocol 1228
   serviceName 1228
   sysID 1228
IPService API 1227
IsJavaObject 1949

J

J2js
   j2js(Object javaObject) 1232
Java Apache Axis2 Web Services Client Examples
   2238, 2239, 2240, 2241
JavaScript debug window
   using 2039
JavaScript engine 2014
JavaScript log 2037
JavaScript Log 2038, 2038
JavaScript modes 2015
JavaScript regular expressions
   Java syntax 1828
javascript tools
   script includes 1947
JavaScriptProbe
   addParameter(String name, String value) 1229
   create() 1229
   JavascriptProbe(String midServer) 1230
   setName(String name) 1230
   setSource(String ip) 1231
JavaScriptProbe API 1229
Jelly 1872
jelly replacement
   add escaping 1929
JSON
   decode(String str) 1233
   encode(Object jsonObject) 1234
   JSON() 1234
   parse(String str) 1234
   stringify(Object jsonObject) 1235, 1237
JSON API 1232
JSON Scoped API 1237
JSONParser
   JSONParser() 1236
JSONParser API 1236
JSUtil
   doesNotHave(Object item) 1238
   escapeAttr(String text) 1238
   escapeText(String text) 1239
   getBooleanValue(GlideRecord, String field) 1239
   host(Object item) 1240
   instance_of(Object Item, String class) 1241
   isJavaObject(Object value) 1241
   logObject(Object obj, String name) 1242
   nil(Object item) 1243
   notNil(Object Item) 1243
   toBoolean(Object item) 1244
   type_of(Object value) 1245
   unescapeAttr(String text) 1246
   unescapeText(String text) 1246
JSUtil API 1238
JSValidator 1949

L

Legacy Notify 1296
limit the number of results
   use filters 2261
Logger
   log(String msg) 1247
   logError(String msg) 1247
   logWarning(String msg) 1248
Logger API 1247
Long-Running SOAP Request Support 2188
LSOFParser
  error(String msg) 1248
  initFileDescriptor() 1249
  initProcess() 1249
  on_c(String line) 1250
  on_endFileDescriptor() 1251
  on_endProcess() 1251
  on_f(String line) 1250
  on_n(String line) 1250
  on_p(String line) 1249
  on_T(String line) 1250
  parse(String lsofOutput) 1251
LSOFParser API 1248

M
Managing Services 274, 276, 276, 276, 276, 277, 278,
maximum number of rows returned specify 2365
MetricBase Time Series API
  GET 1636, 1639
  POST 1643
Microsoft .NET 2244, 2245
  web services client examples 2244, 2245
MIDServer
  getByName(String name) 1252
  getDefault(DiscoverySchedule schedule) 1253
  getForPing(DiscoverySchedule schedule, DiscoveryRange range) 1252
  hostname 1253
  hostOS 1253
  ip 1253
  name 1253
  network 1254
  routerIP 1254
  status 1254
  sysID 1254
  url 1254
  version 1255
  windowsDomain 1255
MIDServer API 1252
MIDServerAjax
  ajaxFunction_testProbe() 1255
  MIDServerAjax() 1256
MIDServerAjax API 1255
MIDServerCluster
  clusterExists() 1256
  getClusterAgent() 1256
  MIDServerCluster(GlideRecord agent) 1257
MIDServerCluster API 1256
MIDServerFileSync
  MIDServerFileSync() 1257
  notifyMIDServers(String table) 1258
MIDServerFileSync API 1257
MIDServerFinder
  findMIDServers() 1258
  getActive() 1260
  setActive(Boolean flag) 1260
  setCapabilities(Array capabilities) 1260
  setDebug(Boolean onOrOff) 1261
  setRanges(String ranges) 1261
  setRangesByIPHostname(String ipOrHostname) 1262
MIDServerFinder API 1258
MIDServerManage
  grab_logs(String agentName, String logs) 1262
  MIDServerManage() 1263
  restart(String agentName) 1263
  stop(String agentName) 1263
  system_msg(String agentName, String cmd, String name) 1264
  text_probe(String agentName, String probeID, String topic, String ename, String source) 1264
  upgrade(String agentName) 1265
MIDServerManage API 1262
MIDServerSelector
  findAgent(String target) 1265
  getError() 1266
  getWarning() 1266
  MIDServerSelector() 1267
  setCapabilities(Array capabilities) 1267
MIDServerSelector API 1265
Mobile Client GlideForm (g form) scripting 1915
Mobile g_form 807, 808, 809, 809
Mobile Glideform 807, 808, 809, 809
Mobile GlideForm 807
MonitorMIDServer
  killOldRequests() 1268
  markDowners() 1268
  monitor() 1269
  MonitorMIDServer() 1269
  sendHeartbeatRequests() 1269
MonitorMIDServer API 1268
move test 124
multi-table processor
    creating 1876
Mutual authentication
    Outbound web services 2325, 2325, 2326
    Protocol profile 2325

N
namespace identifier 7
NameValuePairs
  mapToString() 1270
  NameValuePairs(Object mapForm) 1270
  NameValuePairs(String stringForm) 1270
  quotify(Object value) 1271
  stringToMap() 1271
NameValuePairs API 1270
NetwareHandler
  classifyAndIdentify() 1271
NetwareHandler API 1271
new service categories
  components 272
  Email notifications 272
  forms 272
  service catalog categories 272
tables 272
user roles 272
news
adding
scroller 2286
scroller
adding 2286
news panel
adding
scrolling 2285, 2286
scrolling
adding 2285, 2286
Notify
API 1272
call(String notifyPhoneNumber, String toPhoneNumber, GlideRecord conferenceCall) 1272
call(String notifyPhoneNumber, String toPhoneNumber) 1272
conferenceCall() 1274
dequeueCall(GlideRecord callRecord) 1274
forwardCall(GlideRecord call, String destination, String dtmf) 1275
getAvailableClients(String notifyNumber) 1275
getchildCallIds(GlideRecord callRecord) 1276
getParentCallId(GlideRecord callRecord) 1276
getPhoneNumbers() 1277
getTokens() 1278
getTokens(GlideRecord record) 1278
kick(GlideRecord user) 1279
queueCall(GlideRecord callRecord) 1279
sendSMS(NotifyPhoneNumber notifyPhoneNumber, String toPhoneNumber, String messageBody, GlideRecord source) 1280
sendSMS(NotifyPhoneNumber notifyPhoneNumber, String toPhoneNumber) 1280
NotifyPhoneNumber
getDialCode() 1289
getID() 1289
getNumber() 1289
getOwner() 1290
getTerritory() 1290
supportsConferenceCall() 1290
supportsIncomingPhoneCall() 1291
supportsIncomingSMS() 1291
supportsOutgoingPhoneCall() 1291
supportsOutgoingSMS() 1292
supportsRecording() 1292
supportsWebRTC() 1292
OAuth
OAuthUtil 1320
profile 2299
provider 2298
scope 2298, 2301
third-party provider scenario 2296
OAuthUtil
interceptRequestParameters(requestParamMap) 1320
parseTokenResponse(accessTokenResponse) 1320
OCRIcalSNC
invalidRotaRespCache (GlideRecord rotaRespCacheGR) 1337
OCRosterSpanApprovalUtil
approvePTOSpan (GlideRecord rosterSpanGr) 1321
gerContextualCallUrlPerSpanProposal (GlideRecord rosterSpanProposalGr) 1321
gerFromDateDisplayValuePerSpanProposal (GlideRecord rosterSpanProposalGr) 1322
gerPTOAproversList getPTOAproversList (GlideRecord rosterSpanProposalGr) 1322
gerToDateDisplayValuePerSpanProposal (rosterSpanProposalGr GlideRecord) 1322
getNamePerSpanProposal (GlideRecord rosterSpanProposalGr) 1322
OnCallRotation() 1345
spansOverlap(ArrayList parent, ArrayList child,
String timeZone) 1346
OnCallRotation API 1342
onLoad 2037, 2037
openFrameAPI
hide() 1313
init(Object config, function successCallback,
function failureCallback) 1313
isVisible(function callback) 1314
openCustomURL(Object details) 1314
openFrameAPI(Object details) 1315
openServiceNowForm(Object details) 1315
openServiceNowList(Object details) 1315
setIcons(Array icons) 1316
setSize(Number width, Number height) 1316
setSubtitle(String subTitle) 1317
setTitle(String title) 1317
setTitleIcon(Object icon) 1318
show() 1318
subscribe(openFrameAPIEVENT event, function
eventCallback) 1319
version() 1319

OrderGuide
getID() 1350
init(Map request) 1351
isTwoStep() 1353
navigateFromMap(Map itemDetails) 1354
OrderGuide(String sys_id) 1350
OrderGuideJS
isIncludeItems() 1352
isUseCustomCart() 1353

Outbound REST
authentication 2294
Create a REST message 2292
Define a header 2293
Define an HTTP method 2293
Generate a script preview 2305
generate variables 2304
MID Server 2294
mutual authentication 2304
OAuth 2.0
Google Contacts tutorial 2302
REST message elements 2291
Scripting 2304
Testing HTTP methods 2293
Variable substitution 2304
Outbound SOAP
External XSD file 2315
Outbound SOAP web service
Video tutorial 2308
Outbound SOAP Web Service
basic authentication 2321
configure soap with a proxy 2322
connectivity details 2321
demonstration video 2321
mid server 2314
outbound soap security 2321
soap message 2308, 2308
soap message functions 2311
soap messages 2314
test the soap message 2313
use soap message in a script 2323
variable substitution 2313
web service security 2322
Outbound web services
Mutual authentication 2325, 2325, 2326
output values 108
output variable 135
override execution plan
script 1892

P
packages call
find 2048
packages call removal tool
activate 2047
Packages Call Removal Tool
error types 2077
Packages calls 2052
 PADomainUtils
copy(String runAs) 1355
copyDashboard(String dashboardld, String runAs) 1355
copyJob(String paJob, String runAs) 1356
isWritable(String table, String id) 1356
move(String runAs) 1357
PADomainUtils() 1357
PADomainUtils(String domainFrom) 1358
setFoundation(boolean foundation) 1358
setOverrides(Boolean overrides) 1359

parent instance 307
PAScorecard
addParam(String parameter, String value) 1359
asJSON() 1360
query() 1361
result() 1361
PASnapshot
cgetCompareIDs(String sys_id, Number date1,
Number date2, String type) 1361
getCompareQuery(String sys_id, number date1,
number date2, String type) 1362
getIDs(String sys_id, Number date) 1363
getQuery(String sys_id, number date) 1364

pause script 2021
PDF Web service
parameters 2276
Performance Analytics
API 1681, 1681
Perl
API change 2218
client 2247
eamples 2247
web services 2247
Perl API
approval 2225
architecture 2199
classes 2201
configuration 2222
eamples 2226
install 2195
RequestedItem 2221
SC Task 2224
system requirements 2195
ticket 2224
Perl API attachment 2223
Perl API Dictionary 2221
Perl API GlideRecord 2212
Perl API Incident 2215
Perl API Request 2220
Perl API Task
subroutines 2214
plugin
Enhanced Web Service Provider - Common 2175, 2176
preserve applications 79
preview problems resolving 360
priority 1
example incident 2284
incident example 2284
ProbeHandlerCim
getCimQueries() 1365
getNamespace() 1365
getParameters() 1365
run() 1366
setProbeParameters(Object params) 1365
ProbeHandlerCim API 1364
ProbeHandlerMulti
addParameters(String parent, String sysID, GlideRecord probe) 1366
addProbe(String parent, String sysID) 1367
run() 1367
runProbeHandler(String params, String sysID, GlideRecord probe) 1367
setParameter(String parent, String name, String value) 1368
ProbeHandlerMulti API 1366
ProbeHandlerSNMP
addBoolean(String dbName, String paramName, String paramMap) 1368
addDebug(String paramMap) 1369
addJavascriptHandlers(String paramMap) 1369
addOIDAutoResolve(String paramMap) 1370
addOIDSpec(String paramMap) 1370
addSensorHandler(String paramMap) 1370
addStage(String paramMap) 1370
addTimingSpecs(String paramMap) 1370
getParameters() 1371
run() 1371
setProbeParameters(String params) 1371
ProbeHandlerSNMP API 1368
ProbeHandlerWMI
getParameters() 1372
getWMIFields() 1372
run() 1372
setProbeParameters(Object params) 1373
ProbeHandlerWMI API 1372
Problem Management
Perl API problem 2219
processor
multi-table 1876
processors
API components 1875
creating 1876
secure 1875, 1876
simple 1876
URL endpoints 1873
pull data
local data store 2262
pushes and pulls
Team Development 312
put the class name here 381
Python
client 2252
element 2252
web services 2252
Q
query
test 2360
query tables 1884
queue
get(sys_id) 1373
Queue
join(String) 1374
QuickRanges
createItem(String table, String id, String type) 1374
onMakeRanges() 1375
QuickRanges API 1374
R
received emails
reprocess 1990
RecordToHTML
RecordToHTML(String table, String sys_id, String pattern, Boolean link) 1375
setParameter(String fieldName, String value) 1376
toString() 1376
RecordToHTML API 1375
reference
qualifiers 1895
Reference
qualifiers 1969
reference glide list
from a script 1861, 1862
RenderProperties 1377
RenderProperties
getEncodedQuery() 1377
getListControl() 1377
getParameterValue(String parameterName) 1377
getViewURL() 1378
getViewID() 1378
getViewName() 1378
getWindowProperties() 1379
isDevStudio() 1379
isInteractive() 1379
isManyToMany() 1380
isRelatedList() 1380
resolving
collisions 300, 301
REST 1676, 2294
REST - Inbound 173
REST & SOAP API analytics
  API naming 2289
  collection and cleanup 2288
  dashboards 2287
REST API
  Aggregate 1597
  CORS domain requirements 2093
  CORS support 2092
  define a CORS rule 2093
  Dot-walking 2097
  Example client apps 2090
  explore a table 2090
  explorer 2088
  explorer interface 2088
  getting started 2079
  Performance Analytics 1681, 1681
  Returning session debug logs 2101
  User Role Inheritance 1807
REST API explorer interface 2088
REST message
  Basic auth profile 2295
  Define a header 2293
  elements 2291
  REST message with basic auth 2295
  REST message with OAuth 2296
  Testing HTTP methods 2293
  Variable substitution 2304
REST Message
  Define an HTTP method 2293
  Generate a script preview 2305
  Scripting 2304
REST OAuth
  Google provider tutorial 2302, 2302, 2303
RESTAPIRequest
  body 1384, 2113
  getHeader(String header) 1384, 2116
  getSupportedResponseContentTypes() 1384, 2116
  headers 1385, 2115
  pathParams 1385, 2114
  queryParams 1386, 2114
  queryString 1386, 2114
  url 1386, 2115
  url 1387, 2115
RESTAPIRequestBody
  data 1388, 2117
  dataStream 1389, 2118
  dataString 1389, 2118
  hasNext() 1389, 2119
  nextEntry() 1390, 2119
RESTAPIResponse
  getStreamWriter() 1391, 2120
  setBody(Object body) 1391, 2121
  setContentType(String contenttype) 1392, 2123
  setError(Object error) 1393, 2124
  setHeader(String header, String value) 1393, 2123
  setHeaders(Object headers) 1393, 2122
  setLocation(String location) 1394, 2122
  setStatus(Number status) 1394, 2122
RESTAPIResponseStream
  writeStream(Object stream) 1395, 2124
writeStream(String data) 1396, 2125
RESTMessageV2
  execute() 1397, 1398
  RESTMessageV2() 1400
  RESTMessageV2(String name, String methodName) 1401
  setHttpMethod(String method) 1407
  setRequestorProfile(String requestorContext, String requestorId) 1412
RESTMessageV2
  asynchronous example 2306
  Example direct message 2306
  Example empty message 2308
  Example with MID server 2307
  getEndpoint() 1398
  getRequestBody() 1399
  getRequestHeader(String headerName) 1399
  getRequestHeaders() 1400
  saveResponseBodyAsAttachment(String tableName, String recordSysId, String fileName, String encryptContext) 1401, 1402
  setAuthenticationProfile(String type, String profileId) 1403
  setBasicAuth(String userName, String userPass) 1404
  setEccCorrelator(String correlator) 1405
  setEccParameter(String name, String value) 1406
  setEndpoint(String endpoint) 1406
  setHttpTimeout(Number timeoutMs) 1407
  setLogLevel(String level) 1408
  setMIDServer(String midServer) 1408
  setMutualAuth(String profileName) 1409
  setQueryParameter(String name, String value) 1409
  setRequestBody(String body) 1410
  setRequestBodyFromAttachment(String tableName, String recordSysId, String fileName, String encryptContext) 1410
  setRequestorProfile(String requestorContext, String requestorId) 1410
  setRequestorHeader(String name, String value) 1411
  setStringParameter(String name, String value) 1412
  setStringParameterNoEscape(String name, String value) 1413
RESTResponseV2
  getAllHeaders() 1414
  getBody() 1414
  getCookies() 1415
  getErrorCode() 1415, 1416
  getHeader(String name) 1416
  getHeaders() 1417
  getQueryString() 1417
  getResponseAttachmentSysId() 1418
  getStatusCode() 1418
  haveError() 1419
  waitForResponse(Number timeoutSecs) 1419
  Restricted caller access 37, 38, 39
  retrieve displayed values
    choice list 1923
  retrieve value 108
  Return informative HTTP status code
    Scripted REST APIs 2112
  Return useful error information
    Scripted REST APIs 2112
  role 1972
  rollup 214
  Rosters 1324, 1330
Rotas 1324, 1330
RoundingInterval
  getRoundedValue(Number value) 1420
  getRoundedValue(String type) 1420
RoundingInterval API
  getRoundedValue(Number value) 1420
RoundingInterval(String type) 1420
RSS Feed Generator 2279, 2280, 2281, 2282
RSS Feed Reader 2283, 2286
rules 1849
run automated test 103
runtime access 22, 23, 69

S

Schedule Pages 1950, 1950, 1952, 1953
Schedules 1324, 1330
scoped administration 56, 56
Scoped Application Restricted Caller Access 38
Scoped CartJS
  addToCart(Map request) 448
  CartJS() 449
  CartJS(String cartName) 449
  checkoutCart() 450
  empty() 452
  getCartID() 452
  getCartItems() 453
  getDeliveryAddress() 453
  getRequestedFor() 454
  getRequestedForDisplayname() 454
  getSpecialInstructions() 455
  orderNow(Map request) 456
  setDeliveryAddress(String address) 457
  setRequestedFor(String user) 463
  setSpecialInstructions(String specialInstructions) 458
  submitOrder(Map request) 458
  updateCartItem(Map request, String cart_item_id) 460
Scoped CatalogClientScript
  addScript(String script) 465
  appliesToCatalogItem(Boolean flag) 466
  appliesToCatalogTask(Boolean flag) 466
  appliesToRequestedItem(Boolean flag) 467
  appliesToTargetRecord(Boolean flag) 467
  CatalogClientScript() 468
  create(Boolean standardUpdate) 468
  deleteRecord(String sysID, Boolean standardUpdate) 468
  setAttributes(Map attributes) 469
  setCatalogItem(String sysId) 469
  setOnChangeVariable(String sysId) 470
  setVariableSet(String sysId) 470
Scoped CatalogJS 428
Scoped CatalogSearch
  CatalogSearch() 446
  search(String catalogID, String categoryID, String
  term, Boolean mobile, Boolean depthSearch) 446
Scoped CatCategory
  availableForUserCriteria(String action, Array() criteriaIDs) 474
  notAvailableForUserCriteria(String action, Array() criteriaIDs) 475
  availableForUserCriteria(String action, Array() criteriaIDs) 442
  notAvailableForUserCriteria(String action, Array() criteriaIDs) 442
  }
subtract(GlideDate start, GlideDate end) 613
Scoped GlideDateTime
  add(GlideTime gd) 661
  add(Number milliseconds) 661
  addDaysLocalTime(Number days) 662
  addDaysUTC(Number days) 662
  addMonthsLocaltime(Number months) 663
  addMonthsUTC(Number months) 663
  addSeconds(Number seconds) 664
  addWeeksLocalTime(Number weeks) 664
  addWeeksUTC(Number weeks) 664
  addYearsLocalTime(Number years) 665
  addYearsUTC(Number years) 665
  after(GlideDateTime gdt) 666
  before(GlideDateTime gdt) 666
  compareTo(Object o) 667
  equals(Object dateTime) 668
  getDate() 668
  getDayOfMonthLocalTime() 669
  getDayOfMonthUTC() 669
  getDayOfWeekLocalTime() 670
  getDayOfWeekUTC() 669
  getDaysInMonthLocalTime() 670
  getDaysInMonthUTC() 670
  getDisplayValue() 671
  getDisplayValueInternal() 672
  getDSTOffset() 672
  getInternalFormattedLocalTime() 673
  getLocalDate() 674
  getTime() 674
  getMonthLocalTime() 674
  getMonthUTC() 674
  getNumericValue() 675
  getWeekOfYearLocalTime() 675
  getWeekOfYearUTC() 675
  getYearLocalTime() 675
  getYearUTC() 675
  GlideDateTime() 676
  GlideDateTime(GlideDateTime g) 676
  GlideDateTime(String value) 676
  hasDate() 680
  isDST() 681
  isValid() 681
  onOrAfter(GlideDateTime gdt) 681
  onOrBefore(GlideDateTime gdt) 682
  setDayOfMonthLocalTime(Number day) 682
  setDayOfMonthUTC(Number day) 683
  setDisplayValue(String asDisplayed) 683
  setDisplayValue(String value, String format) 684
  setDisplayValueInternal(String value) 685
  setDSTOffset(Number offset) 686
  setValue(Object value) 686
  setValueUTC(String dt, String format) 686
  setYearLocalTime(Number year) 688
  setYearUTC(Number year) 689
subtract(GlideDateTime start, GlideDateTime end) 690
subtract(GlideTime time) 690
subtract(Number milliseconds) 690
toString() 691
Scoped GlideDBFunctionBuilder
  add() 692
  build() 692
  concat() 693
  constant(String constant) 693
  datediff() 694
  dayOfWeek() 695
  divide() 696
  field(String field) 696
  GlideDBFunctionBuilder() 697
  length() 697
  multiply() 697
  subtract() 698
Scoped GlideDigest
  getMD5Base64(String source) 703
  getMD5Base64FromInputStream(GlideScriptableInputStream inputStream) 703
  getMD5Hex(String source) 703
  getMD5HexFromInputStream(GlideScriptableInputStream inputStream) 704
  getSHA1Base64(String source) 704
  getSHA1Base64FromInputStream(GlideScriptableInputStream inputStream) 705
  getSHA1Hex(String source) 705
  getSHA1HexFromInputStream(GlideScriptableInputStream inputStream) 706
  getSHA256Base64(String source) 706
  getSHA256Base64FromInputStream(GlideScriptableInputStream inputStream) 707
  getSHA256Hex(String source) 707
  getSHA256HexFromInputStream(GlideScriptableInputStream inputStream) 708
  GlideDigest() 708
Scoped GlideDuration
  add(GlideDuration duration) 709
  getByFormat(String format) 710
  getDayPart() 710
  getDisplayValue() 711
  getDurationValue() 711
  getRoundedDayPart() 712
  getValue() 712
  GlideDuration() 711
  GlideDuration(GlideDuration another) 713
  GlideDuration(Number milliseconds) 713
  GlideDuration(String displayValue) 713
  setDisplayValue(String asDisplayed) 713
  setValue(Object o) 714
  subtract(GlideDuration duration) 714
Scoped GlideElement
  changes() 740
  changesFrom(Object o) 741
  changesTo(Object o) 742
  getChoices(Object o) 743
  getChoiceValue() 744
  getChoiceValue() 744
  getDisplayValue(Number maxCharacters) 746
  getED() 746
  getGlobalDisplayValue() 747
getCatalogItem(String sysId) 1025
gField(String fieldName) 1028
gField(String record, String fieldName) 1028
gField(Object record, String fieldName) 1029
gField(Object record, String fieldName) 1029
gForm(String tableName, String sysId) 1029
gKBCategoryArticless() 1030
gKBCategoryArticleSummaries(String sysId, Number limit, Number maxChars) 1030
gKBCount(String sysId) 1031
gListColumns(String tableName, String view) 1032
gMenuHref(GlideRecord page) 1033
gMenuItems(String sysId) 1032
gParameter(String name) 1033
gPortalRecord() 1033
gRecord() 1034
gRecordDisplayValues(Object data, GlideRecord from, String names) 1035
gRecordElements(Object data, GlideRecord from, String names) 1035
gRecordValues(Object data, GlideRecord from, String names) 1036
gStream(String table, String sysId) 1040
gUserInitials() 1040
getValue(String name) 1041
getValues(Object data, String names) 1041
getWidget(String sysId, Object options) 1043
mapUrlToSPUrl(String url) 1044
Scoped GlideSysAttachment
copy(String sourceTable, String sourceID, String targetTable, String targetID) 1045
deleteAttachment(String attachmentId) 1045
getAttachments() 1045
gGetContent(GlideRecord sysAttachment) 1046
gGetContentBase64(GlideRecord sysAttachment) 1047
gGetInputStream(GlideRecord sysId) 1047
GlideSysAttachment() 1047
write(GlideRecord record, String fileName, String contentType, String content) 1047
writeBase64(GlideRecord gr, String fileName, String contentType, String content_base64Encoded) 1048
writeContentStream(GlideRecord gr, String fileName, String contentType, GlideScriptableInputStream inputStream) 1048
Scoped GlideSysListControl
getControlID() 1049
GlideSysListControl(String tableName) 1049
isOmitEditButton() 1049
isOmitNewButton() 1050
Scoped GlideSystem
addErrorMessage( Object message) 1103
addInfoMessage( Object message) 1104
base64Decode(String source) 1105
base64Encode(String source) 1105
dateGenerate(String date, String range) 1108
debug( String message, Object parm1, Object parm2, Object parm3, Object parm4, Object parm5) 1110
doUnWeek() 1112
error(String message, Object parm1, Object parm2, Object parm3, Object parm4, Object parm5) 1114
eventQueue(String name, Object instance, String parm1, String parm2, String queue) 1115
eventQueueScheduled(String name, Object instance, String parm1, String parm2, Object expiration) 1116
executeNow(GlideRecord job) 1116
generateGUID() 1116
generateGUID() 1116
generateGUID() 1116
gCallerScopeName() 1117
gCssCacheVersion(String) 1118
gCurrentApplicationId() 1118
gCurrentScopeName() 1119
gEscapedMessage(String id, Object args) 1120
gMessage(String id, Object args) 1120
getProperty(String key, Object alt) 1121
getSession() 1121
getSessionId() 1122
getSessionToken() 1122
gTimeZoneName() 1123
getUrlOnStack() 1123
getUser() 1123
getUserDisplayName() 1124
getUserID() 1124
getUserInitials() 1124
getUserPermission(String name) 1124
isInteractive() 1124
isImpersonating() 1124
isOmitNewButton() 1124
isOmitEditButton() 1124
isReadonly() 1124
isReadOnly() 1124
getUserName() 1125
hasRole(Object role) 1125
include(String name) 1127
info(String message, Object parm1, Object parm2, Object parm3, Object parm4, Object parm5) 1127
isDebugging() 1128
isInteractive() 1129
isLoggedln() 1129
isMobile() 1129
nil(Object o) 1131
setProperty(String key, String value, String description) 1132
setRedirect(Object o) 1133
tableExists(String name) 1133
urlDecode(String url) 1134
urlEncode(String url) 1134
warn(String message, Object parm1, Object parm2, Object parm3, Object parm4, Object parm5) 1134
xmlToJSON(String xmlString) 1135
yearsAgo(Number years) 1136
getAllExtensions() 1137
getBase() 1137
getHierarchy() 1138
getName() 1138
getRoot() 1138
gGetTableExtensions() 1139
gGetTables() 1139
GlideTableHierarchy(String tableName) 1140
hasExtensions() 1141
isBaseClass() 1140
isSoloClass() 1141
getEncoding() 1142
GlideTextReader(GlideScriptableInputStream inputStream) 1142
readLine() 1142
getByFormat(String format) 1143
gDisplayValue() 1143
gDisplayValueInItem() 1144
gHourLocalTime() 1144
gHourOfDayLocalTime() 1145
gHourOfDayUTC() 1145
gHourUTC() 1145
gMinutesLocalTime() 1146
gMinutesUTC() 1146
gSecond() 1146
gValue() 1147
GlideTime() 1147
GlideTime(Number milliseconds) 1147
setDisplayValue(String asDisplayed) 1148
setvalue(String o) 1148
subtract(GlideTime startTime, GlideTime endTime) 1149
getCompanyID() 1177
gDisplayName() 1178
gEmail() 1178
getFirstName() 1179
gID() 1179
getLastname() 1180
gName() 1180
gPreferene(String name) 1181
gRoles() 1181
getUserRoles() 1181
hasRole(String role) 1182
isMemberOf(String group) 1182
savePreference(String name, String value) 1183
getEncodedQuery() 1380
getListControl() 1381
getParameterValue(String parameterName) 1381
getReferringURL() 1381
getViewName() 1382
getWindowProperties() 1382
isDevStudio() 1382
isInteractive() 1383
isManyToMany() 1383
isRelatedList() 1383
space(Number spaces) 1490
execute(GlideDateTime start, GlideDateTime end) 1508
groupBy(String field) 1509
metric(String metricName) 1509
create(Object columnValues, Boolean standardUpdate) 1523
deleteRecord() 1524
read(String[] columns) 1525
setActive(Boolean active) 1525
setAdvanced(boolean advanced) 1526
setCompanies(String companies) 1526
setDepartments(String departments) 1527
setGroups(String groups) 1527
setLocations(String locations) 1528
setMatchAll(Boolean match_all) 1528
setName(String name) 1529
setRoles(String roles) 1529
setScript(String script) 1529
setUsers(String users) 1530
update(Object columnValues, String reason) 1531
UserCriteria() 1531
UserCriteria(String sys_id) 1531
debug(String message, Object args) 1557
error(String message, Object args) 1557
Scoped GlideTableHierarchy
Scoped GlideTextReader
Scoped GlideTime
Scoped GlideUICompatibility
Scoped GlideUser
Scoped GlideURI
Scoped IdentificationEngine
Scoped JSON
Scoped OrderGuide
Scoped Queue
Scoped RenderProperties
Scoped ScopedRenderProperties
Scoped TemplatePrinter
Scoped Transformer
Scoped UserCriteria
Scoped Workflow
4212
getVariable(String name) 1558
info(String message, Object args) 1558
inputs 1559
name() 1559
removeVariable(String name) 1559
result 1560
scratchpad() 1560
setResult(String result) 1561
setVariable(String name, Object value) 1561
warn(String message, Object args) 1561

Scoped XMLDocument2
createElement(String name) 1577
createElementWithTextValue(String name, String value) 1577
gеДocumentElement() 1578
getFirstChild(String xPath) 1579
getFirstChild(Object current) 1580
getNode(String xPath) 1580
getNodeText(String xPath) 1581
parseXML(String xmlDoc) 1582
setCurrentElement(XMLNode element) 1582
setNameAware(Boolean aware) 1583
toString() 1583
XMLDocument2(GlideScriptableInputStream inputSteam) 1584
XMLDocument2() 1584

Scoped XMLNode
gетAttribute(String attribute) 1584
getAttributes() 1585
getChildNodeIterator() 1585
getFirstChild() 1586
getLastChild() 1587
getNodeName() 1587
getNodeValue() 1588
getTextContent() 1589
hasAttribute(String attribute) 1589
toString() 1590

Scoped XMLNodeIterator
hasNext() 1591
next() 1591

Scoped, Global HistoryWalker 1191
script 1863
script access 22, 23
script API
  service catalog 1898
script debugger
  User interface 2016
Script Debugger status 2021
script field highlighting 2040
script include
  AJAXClientTiming 1923
  GenericException 1948
  IsJavaObject 1949
  JSValidator 1949
script includes
  protection policy 52
Script Includes
  change privacy 1866, 1867
  client-callable 1866, 1867
  privacy settings 1866
script syntax error
  check 1837
Scriptable Service Catalog Variables 1897
Scripted REST API
  core concepts 2102
  streaming vs large objects example 2131
Scripted REST APIs
  Access controls 2104
  add a version 2109
  Available error objects 2105
  Build tests verify functionality 2113
  content type 2110
  create 2107
  create a resource 2107
  Define query parameters 2108
  database versioning 2109
  Enforce and test access controls 2113
  Error objects 2104
  Examples 2126
  Explore 2111
  Follow REST API conventions 2111
  Maximum request size 2110
  Require resource ACL 2109
  RESTAPIRequest 1384, 2113
  RESTAPIRequestBody 1387, 2116
  RESTAPIResponse 1391, 2120
  RESTAPIResponseStream 1395, 2124
  Return Informative HTTP status code 2112
  Return useful error information 2112
  roles 2103
  script samples 2126
  security 2104
  Sending error objects 2105
  service headers 2108
  streaming file attachments example 2135
  URI format 2102
  Use versioning 2112
  versioning 2103
scripted web service 2234, 2234
scripting
  API suggestions 1901
  business rules 1862
  messages 1940
Scripting
  business rules 2030
  debug 2030
ScriptLoader
  getScripts(Array scripts, Function callback) 1421
  getScripts(String filePath, Function callback) 1421
ScriptLoader API 1420
  scripts 1878
Scripts - Background 2026
Scrollable areas 2283
scrolling
  adding
    news 2286
    news
    adding 2286
scrolling
  adding
    elements 2284
    news panel 2285, 2286

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
SNMPNetworkInterfaces
process(String cmdb_ci, String doc, Boolean isComputer, String agentName, String ecc_sys_id) 1436
SNMPNetworkInterfaces API 1436
SNMPResponse
buildIndex(Object cur, String name, String extName) 1436
buildReferences() 1437
getAmbiguousOIDs(String refBase, String refLeaf) 1437
getOIDInt(String name) 1437
getOIDObj(String name) 1438
getOIDTable(String parentName, String entryName) 1438
getOIDText(String name) 1439
normalize(Object cur) 1439
SNMPResponse API 1436
SOAP
Outbound web service 2308
Retrieve a large number of records 2261
security 2164
web service 2136, 2154, 2157, 2163, 2163, 2181, 2182, 2182, 2182
SOAP default security policy 2178
SOAP message
hierarchical 2186
SOAP requests, signed certificates required 2176
SOAP session
management 2187
reporting 2187
SOAP session properties 2187
SOAP session report
view 2188
SOAP strategies 2261
SOAP web service
FAQ 2190
PERL script, example 2190
security 2190
SOAP message, example 2190
SOAP Web Services
scripted web services 2228, 2233
SOAPMessageV2
execute() 1439
executeAsync() 1440
SOAPMessageV2
Asynchronous message example 2323
Direct message example 2323
getEndpoint() 1440
getRequestBody() 1441
getRequestHeader(String headerName) 1441
getRequestHeaders() 1442
MID Server example 2324
Recordless message example 2325
setBasicAuth(String userName, String userPass) 1442
setEccCorrelator(String correlator) 1443
setEccParameter(String name, String value) 1443
setEndpoint(String endpoint) 1444
setHttpTimeout(Number timeoutMs) 1445
setLogLevel(String level) 1445
setMIDServer(String midServerName) 1446
setMutualAuth(String profileName) 1446
setRequestBody(String requestBody) 1447
setRequestHeader(String headerName, String headerValue) 1447
setSOAPAction(String soapAction) 1448
setStringParameter(String name, String value) 1448
setStringParameterNoEscape(String name, String value) 1449
setWSSecurity(String keystoreId, String keystoreAlias, String keystorePassword, String certificateId) 1449
SOAPMessageV2() 1450
SOAPMessageV2(String soapMessage, String soapFunction) 1450
SOAPResponseV2
getAllHeaders() 1451
getBody() 1451
getCookies() 1452
ger#errorCode() 1452
ger#errorMessage() 1453
ger>getHeader(String name) 1453
getHeaders() 1454
ger>getStatusCode() 1454
haveError() 1455
waitForResource(Number timeoutSecs) 1455
source control
apply remote changes 260
available operations 255
branch 262
commit changes 260
import application 258
integration 254
link application 256
switch branch 263
view commit history 264
spAriaUtil
sendLiveMessage(String message) 1456
spModal
alert(String message).then(fn) 1456
confirm(String message).then(fn) 1457
open(Object options).then(fn) 1458
prompt(String message, String default).then(fn) 1461
spUtil
addErrorMessage(String message) 1462
addInfoMessage(String message) 1462
addTrivialMessage(String message) 1463
format(String, Object) 1464
get(String widgetId Object data) 1463
recordWatch(Object $scope, String table, String filter, Function callback) 1465
refresh(Object $scope) 1464
update(Object $scope) 1465
SQL Server
2008 2367, 2367
2012 2367, 2367
Linked Server 2367
StandardCredentialsProvider
getCredentialByID() 1467
getCredentials() 1467
StandardCredentialsProvider() 1467
StartDiscovery
   addAdditionalIP(String status, String ip, String sensor) 1468
   discoverFromSchedule(String ip, Boolean samePhase, String sensor) 1468
   getCancelScript() 1469
   scheduleCancelJob() 1469
   startFromIP(DiscoverySchedule schedule, String ip) 1469
   startFromSchedule(GlideRecord schedule, GlideRecord job) 1470
StartDiscovery API 1468
stashes 261, 261
state change form color 1974
StateManagementScriptableApi
   addBulkCIAction(String requestorId, String sysIdList, String ciActionName, String ciActionListOld, String leaseTime) 1470
   extendCIActionLease( String requestorId, String ciSysId, String ciActionName, String leaseTime) 1471
   getCIActions(String ciSysId) 1472
   getOperationalState(String ciSysId) 1472
   isCompatibleCIAction(String actionName, String otherActionName) 1472
   isLeaseExpired(String requestorId, String ciSysId, String ciActionName) 1473
   isNotAllowedAction(String ciType, String opsLabel, String actionName) 1473
   isNotAllowedOpsTransition(String ciType, String opsLabel, String transitionOpsLabel) 1474
   isValidRequestor(String requestorId) 1474
   registerOperator() 1475
   removeBulkCIAction(String requestorId, String sysIdList, String ciActionName) 1475
   setBulkCIOperationalState(String requestorId, String sysIdList, String ciActionName, String opsLabel, String opsStateListOld) 1476
   unregisterOperator( String requestorId) 1477
StateManagementScriptableApi API 1470
static WSDL 2234, 2234, 2235
Static WSDL 2233, 2234, 2234, 2235
step description script 236
step in 2019
step out 2019
step over 2019
step-through script 2019
StopWatch
   getTime() 1477
   restart() 1477
   StopWatch() 1478
   StopWatch(Date initialDate) 1478
   toString() 1478
StopWatch API 1477
Studio
   automatic recovery 253
draft records 253
subscriptions enforcing 57
SummaryTableWriter
generate() 1479
setQuery(String query) 1479
setStackOn(String column) 1479
setTitle(String title) 1480
setTrend(String field, String interval) 1480
SummaryTableWriter(String tableName, String column) 1481
syntact editor editing JavaScript scripts 1832
keyboard shortcuts 1838
navigate to a line number 1832
script macros 1830
syntact editor plugin 1830
sys_id 2249
system clone 79
system update sets 328
T
Table API
   DELETE 1765
   explore a table 2090
   GET 1767
   GET id 1776
   PATCH 1783
   POST 1792
   PUT 1799
table cleanup 123, 124
tables
   access settings 24
TableUtils
   drop(String tableName) 1482
   dropAndClean(String tableName) 1483
   dropTableAndExtensions(String tableName) 1483
   getAbsoluteBase() 1485
   getAllExtensions() 1485
   getHierarchy() 1486
   getTableExtensions() 1487
   hasExtensions() 1487
   isBaseClass() 1487
   isSoloClass() 1488
   tableExists() 1482
   TableUtils(String tableName) 1489
TableUtils API 1482
tag
   Glide 1924
   Jelly 1924
Task Active State Management business rule 1858
TaskStateUtil
   ATTR_DEFAULT_CLOSE 1503
   ATTR_DEFAULT_WORK 1503
   ATTR_INACTIVE_STATES 1503
   getDefaultCloseState() 1503
   getDefaultWorkState() 1504
   getInactiveStates() 1504
   isStateInactive(String state) 1505
   runMarkClosed 1505
   runTaskCloser 1506
   runTaskReopener 1506
   setDefaultWorkState(String defaultWorkState) 1506
   SYSTEM_DEFAULT_CLOSE 1506
   SYSTEM_DEFAULT_WORK 1507
SYSTEM_INACTIVE_STATES 1507
TaskStateUtil(GlideRecord task) 1507
TaskStateUtil API 1502
Team Development
access rights 304
approve or reject push 284
back out 285, 297
best practice 280
change the parent instance 287
code review 309
code review request 286
Compare to Current 319
call to peer instances 300
concept 310, 310, 311, 312, 322
dashboard 283
enable code review 307
exclusion policy 304
ignore 291
instance hierarchy 308
local changes 281, 282, 285, 291, 297, 297
local version 290
parent instance 307
process 312
pull a version 293
pull exceptions 282
push 297
push version 294
pushed change 287
pushed version 290
pushes and pulls 312
reconcile 299
remote instance 305
resolve collisions 300
resolve multiple collisions 303
review status 287
roles 313
setting up 303, 308
using 280
versions 313, 313, 314, 315, 321, 321
TemplatePrinter
print(String string) 1489
test scripts 235
test service catalog 195
test template 108, 122
test templates 122
Time
validate 2007
TimelineSpan
addPredecessor() 1490, 1491
getStartTimeMs() 1496
getSysId() 1496
getTable() 1496
getTooltip() 1497
setAllowXDragLeft() 1497
setAllowXDragRight() 1497
setAllowXMov() 1498
setAllowYMov() 1498
setAllowYMovePredecessor() 1499
setInnerSegmentClass() 1499
setInnerSegmentTimeSpan() 1499
setPointIconClass() 1500
setSpanColor() 1500
setSpanText() 1501
setTimeSpan() 1501, 1501
setTooltip() 1502
tracked customizations 328
transaction details 2024
transactions 2024
TransformPart
add(Number constant) 1511
avg() 1511
bottom(Number count) 1511
ceil(Number ceiling) 1512
collect() 1512
count() 1512
div(Number constant) 1513
filter(Object aggregator, Object duration) 1513
floor(Number floor) 1513
fractiles(Array fractions) 1514
getResult() 1514
groupBy(String field) 1515
interpolate(Object count) 1515
label(String label) 1516
limit(Object count) 1516
log(Number base) 1517
max() 1517
median() 1517
metric(String metric) 1518
min() 1518
mul(Number constant) 1518
partition(String aggregator, GlideDateTime duration, GlideDateTime base) 1519
resample(Number count) 1519
round(Number precision) 1520
stddev() 1520
sub(Object constant) 1520
sum() 1521
top(Object constant) 1520
toArray() 1521
tutorial 2259
TransformResult
byGroup() 1522
getByLabel(String label) 1522
data() 1522
toArray() 1523
tutorial 2259
U
UI macros 1872
UI pages 1870, 1871
UI Pages 1868

© 2020 ServiceNow, Inc. All rights reserved.
ServiceNow, the ServiceNow logo, Now, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc., in the United States and/or other countries. Other company names, product names, and logos may be trademarks of the respective companies with which they are associated.
UI routing action
create 1974
UI scripts
create 1903
run 1904
Unencrypted Single Sign-On
ASP script 2006
Update Set picker
granting access 334
update sets
administration 333
backing out 365
best practices 337
collision resolution 350
committing 365
compare 348
completing 352
creating 338
deleting 367
getting started 330
introduction 327
load from XML 353
merging 346
navigating between records 345
overwriting during upgrade 337
previewing 356
procedures 338
reports 346
retrieving 355
viewing for an object 313
Versions
version records 315
view progress automated test 105
VMUtils
turnCorrelationIdToUuid(String correlationId) 1534
turnUuidToCorrelationId(String uuid) 1535
VMUtils API 1534
W
Walkworkflow
computeFullSequences() 1535
computeSequences() 1536
dump() 1536
dumpSequences() 1536
dumpShortestPathOrders() 1537
getActivitiesAtOrder() 1537
getSequences() 1537
walk() 1538
WalkWorkflow() 1538
WalkWorkflow 1535
Web service
security 2286
Web Service
JSON v2 2265, 2265, 2268, 2269, 2269, 2270, 2272, 2273
web service API
aggregate 2142
API functions 2137
Data Modification API 2146, 2146, 2148, 2151, 2152, 2153
Data Retrieval API 2138, 2138, 2139, 2141, 2142
deleteMultiple 2153
deleteRecord 2152
get 2141
getKeys 2138
getRecords 2139
insert 2146
insertMultiple 2148
SOAP API 2137
update 2151

User interface elements 9
user object
get 1969
User Role Inheritance API
GET 1807
user roles 9, 124, 128, 128, 206
user session 2021

V
VariablePoolQuestionSetJS
getFlatQuestions() 1532
load() 1533
setCartID(String id) 1534
VariablePoolQuestionSetJS() 1532
versions
Compare to Current 319
revert a change 321
suppress 321
transferring 314
viewing for an object 313
Versions
version records 315
view progress automated test 105
VMUtils
turnCorrelationIdToUuid(String correlationId) 1534
turnUuidToCorrelationId(String uuid) 1535
VMUtils API 1534
W
Walkworkflow
computeFullSequences() 1535
computeSequences() 1536
dump() 1536
dumpSequences() 1536
dumpShortestPathOrders() 1537
getActivitiesAtOrder() 1537
getSequences() 1537
walk() 1538
WalkWorkflow() 1538
WalkWorkflow 1535
Web service
security 2286
Web Service
JSON v2 2265, 2265, 2268, 2269, 2269, 2270, 2272, 2273
web service API
aggregate 2142
API functions 2137
Data Modification API 2146, 2146, 2148, 2151, 2152, 2153
Data Retrieval API 2138, 2138, 2139, 2141, 2142
deleteMultiple 2153
deleteRecord 2152
get 2141
getKeys 2138
getRecords 2139
insert 2146
insertMultiple 2148
SOAP API 2137
update 2151

User interface elements 9
user object
get 1969
User Role Inheritance API
GET 1807
user roles 9, 124, 128, 128, 206
user session 2021

V
VariablePoolQuestionSetJS
getFlatQuestions() 1532
load() 1533
setCartID(String id) 1534
VariablePoolQuestionSetJS() 1532
versions
Compare to Current 319
revert a change 321
suppress 321
transferring 314
viewing for an object 313
Versions
version records 315
view progress automated test 105
VMUtils
turnCorrelationIdToUuid(String correlationId) 1534
turnUuidToCorrelationId(String uuid) 1535
VMUtils API 1534
W
Walkworkflow
computeFullSequences() 1535
computeSequences() 1536
dump() 1536
dumpSequences() 1536
dumpShortestPathOrders() 1537
getActivitiesAtOrder() 1537
getSequences() 1537
walk() 1538
WalkWorkflow() 1538
WalkWorkflow 1535
Web service
security 2286
Web Service
JSON v2 2265, 2265, 2268, 2269, 2269, 2270, 2272, 2273
web service API
aggregate 2142
API functions 2137
Data Modification API 2146, 2146, 2148, 2151, 2152, 2153
Data Retrieval API 2138, 2138, 2139, 2141, 2142
deleteMultiple 2153
deleteRecord 2152
get 2141
getKeys 2138
getRecords 2139
insert 2146
insertMultiple 2148
SOAP API 2137
update 2151
serviceNow    Kingston    Index

web services
ACL 2164
basic authentication 2165, 2165
internal integration communications 2168
internal integration users 2168
mutual authentication 2181
role requirements 2164
security 2163, 2165, 2165, 2168
SOAP envelope header 2173
SOAP Security Policies 2175
strict security 2180, 2181
UsernameToken 2172
WS Security profile 2168
X.509 token 2171

Web services
Available web services 2078

Web Services
outbound soap web service 2308, 2308, 2308, 2311, 2313, 2313, 2314, 2321, 2321, 2321, 2321, 2322, 2322, 2323

Web Services C Sharp .NET End to End Tutorial
call web service 2256
configuration 2256
results 2260
source code 2259
troubleshoot 2261
use a service reference 2258
use a web reference 2259
visual studio .net 2256

Web services integrations 9

Web Services-Security (WSS) 2177

WFActivityHandler
db() 1539, 1539
generate() 1539
info() 1540, 1540
js() 1541
onCancel() 1541
onExecute() 1541
runScript() 1542
setActivityOutput() 1542
setResultFailed() 1542
setResultSucceeded() 1543
warn() 1543, 1543

WFActivityHandler 1538

WindowsOSNameHelper
formatWindowsOSName(String name) 1544
makeOSChoiceValid(String value) 1544
osCleanupName(String name) 1544

WindowsOSNameHelper API 1544

wizard script
examples 1899
record generators 1900
wizard scripts 1899
Wizard UI policy 1900

Workflow
broadcastEvent() 1545
cancel() 1545
cancelContext() 1546
deleteWorkflow() 1547
fireEvent() 1547
fireEventByld() 1548
getContexts() 1548
getEstimatedDeliveryTime() 1549
getEstimatedDeliveryTimeFromWFVersion() 1549
getReturnValue() 1550
getRunningFlows() 1550
getVersion() 1551
getVersionFromName(String workflowName) 1551
hasWorkflow() 1552
restartWorkflow() 1553
runFlows() 1553
startFlow() 1554
startFlowFromContextInsert() 1555
startFlowRetroactive() 1555

workflow scratchpad
access from business rules 1969

WorkflowDuration
addSeconds() 1562
calculate() 1562
calculateTimeLeft() 1563
gerEndDateTime() 1563
getSeconds() 1563
getTotalSeconds() 1564
setEndDateTime() 1564, 1564
setStartDateTime() 1565, 1565
setUsedSecs() 1565
setWorkflow() 1566

WorkflowModelManager
getActivityHistoryRecordByld() 1567, 1574
getAllTransitionedIntoActivity() 1567
getAllTransitionedIntoActivityId() 1567
getExecutedHistory() 1568
getFinalExecutedActivityIdList() 1569
getFinalExecutedActivityList() 1569
getNextByExecutedOrder() 1570
getNextByExecutedOrderId() 1570
getNextByTransitionId() 1571
getPreviousByExecutedOrder() 1572
getPreviousByExecutedOrderId() 1572
getPreviousByTransition() 1573
getPreviousByTransitionId() 1574

WorkflowModelManagerAjax
getExecutedHistory() 1575
getFinalExecutedActivityIdList() 1575
getRolledBackActivityIdList() 1575

WorkflowScheduler
run() 1576

WorkflowTimeline 1576

WS-security
error logs 2178
error messages 2178
security policy 2177

WSDL Schema
hierarchical data 2184

X

XML web service
parameters 2278
- **XMLDocument Script Object**: 1964, 1967
- **XMLUtilJS**
  - `escapeForXMLText(String text)` 1593
  - `stringToValue(String str)` 1593
  - `unescapeForXMLText(String text)` 1593
  - `valueToString(String XMLvalue)` 1593
- **XMLUtilJS API**: 1592