Geneva ServiceNow Custom Application Development

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.

This PDF was created from content on docs.servicenow.com. The web site is updated frequently. For the most current ServiceNow product documentation, go to docs.servicenow.com.
## Contents

**Custom Application Development** .............................................................................................................. 3  
  Applications......................................................................................................................................................... 3  
    Creating applications........................................................................................................................................ 3  
    Parts of an application.................................................................................................................................. 22  
    Contextual development environment........................................................................................................ 48  
    Application management.............................................................................................................................. 56  
    ServiceNow Studio...................................................................................................................................... 64  
**Service Creator** .............................................................................................................................................. 87  
  Service creator process................................................................................................................................... 87  
  Activate Service Creator.................................................................................................................................. 88  
  Manage a service............................................................................................................................................... 95  
  Add a question.................................................................................................................................................. 100  
  Designing Services............................................................................................................................................. 101  
  Create or edit an unpublished service........................................................................................................... 105  
  Questions.......................................................................................................................................................... 106  
  Publish a service............................................................................................................................................... 106  
**Update Sets** .................................................................................................................................................. 106  
  Getting started with Update Sets................................................................................................................... 106  
  Update Set transfers to other instances......................................................................................................... 112  
  System Update Sets........................................................................................................................................ 113  
  Update Set administration............................................................................................................................... 116  
  Create an Update Set....................................................................................................................................... 118  
  Retrieve an Update Set..................................................................................................................................... 130  
  Merge Update Sets.......................................................................................................................................... 141  
**Team Development** ....................................................................................................................................... 143  
  Team Development overview.......................................................................................................................... 143  
  Team Development setup............................................................................................................................... 168  
  Code reviews.................................................................................................................................................... 175  
  Code review notifications............................................................................................................................... 176  
  Code review workflow.................................................................................................................................... 176  
  Exclusion policies.............................................................................................................................................. 177  
  Instance hierarchies........................................................................................................................................... 178  
  Pulls and pushes............................................................................................................................................... 178  
  Team Development roles.................................................................................................................................. 178  
  Versions............................................................................................................................................................ 178  
  Versions and local changes............................................................................................................................. 185  
**Index** .................................................................................................................................................................. 188
Custom Application Development

Create, manage, and distribute custom applications for your business. You can create custom applications to extend existing functionality delivered with the core platform or introduce new functionality.

Applications

Applications represent packaged solutions for delivering services and managing business processes. The ServiceNow platform offers several standard applications such as Incident, Problem, and Change. Administrators can also develop and manage custom applications to meet business needs. When building applications on the ServiceNow platform, application developers can take advantage of existing platform features such as security access controls, workflow, reporting, and notifications as well as existing public data such as user and task records.

The System Applications > Applications module lists the apps and Update Sets on your ServiceNow instance.

- The Develop tab lists custom apps on your instance. In addition, you can use the tab to create and update (edit) custom apps.
- The Downloads tab lists ServiceNow Store apps and Remote apps that were not created by your organization.
- The Updates tab lists updates sets that are available for the apps on the instance.

Note: The system determines whether an app is created by your organization by comparing the vendor_prefix of the app with the vendor_prefix of the instance. The vendor_prefix is determined by the glide.appcreator.company.code sys_properties record (as described in Namespace identifier on page 50).

Creating applications

The ServiceNow 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 on page 64
  - Application scope on page 48
  - Script protection policy on page 56
- Platform features
  - Security access controls
  - Workflow
  - Business rules
  - APIs
  - Notifications
- Publicly accessible data
  - User records
  - Task records
Geneva    ServiceNow    Custom Application Development

- 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.

Application development process

1. Define the business requirements. Determine the problem you want to solve and how the application meets that need. Answer the questions:
   - What does the application need to do?
   - Who will use it?
   - How will you know if it works?

2. Define the data model. Identify general categories of information you need to track.
   - What data is being captured?
   - What tables are required, and how will they reference each other?
   - What changes do you anticipate later, and does your data model scale to meet these changes?

3. Build the application.
   1. Select an application creation method to create a custom application record and set the application scope.
   2. Create application data tables to store application-specific data.
   3. Design the user interface, such as the list and form layout.
   4. Set application access settings to permit or restrict other applications from accessing application data.
   5. Add business logic and automation such as business rules, UI policies, notifications, and workflows, to meet your business requirements.

4. Test the application. Locate the application in the navigator and make sure it works as you expect. Engage users early in the testing process. Document their feedback and make sure they are satisfied before moving to the next step.

5. Share the application. When it passes testing, you can share the application with other instances by one of these methods.
   - Publish the application to the ServiceNow application repository to share it with other other instances assigned to the same company.
   - Publish the application to the ServiceNow Store to share it with everyone.
   - Publish the application as an Update Set to comply with a change management or backup policy.
   - Push the application to other instances in the team development environment.

Application creation options

Application developers have options when creating applications.
Figure 1: Sample application creation options

To start creating an application, navigate to System Applications > Applications and click New. 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 (available only when a system property is set)

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.
Create a custom application

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 sub-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. 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. You can view a log of changes from completed conversions by navigating to **System Applications > Service Application Log**.

<table>
<thead>
<tr>
<th>This 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**

Set the system property glide.app.creator.global to **true**.

Applications created in the global scope do not benefit from application scope protections. This option is intended for those who need to support legacy applications from previous versions.

1. From the Start from global 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>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</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 use the application.

5. Click **Create**.

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

### Application data model and logic

Most applications need both a data model and application logic.

Depending on the creation option you selected, one or more of these components may already exist. You can use the custom application record to see and navigate to these elements.

**Note:** The application you select from the application picker determines both where and whether your changes are saved.

### 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.
**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.

<table>
<thead>
<tr>
<th>Action</th>
<th>Access granted</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a list view</td>
<td>Always allowed</td>
<td>This action is always available to users with access to customization.</td>
</tr>
<tr>
<td>Create a form view</td>
<td>Always allowed</td>
<td>This action is always available to users with access to customization.</td>
</tr>
<tr>
<td>Create a form section</td>
<td>Always allowed</td>
<td>This action is always available to users with access to customization.</td>
</tr>
<tr>
<td>Select fields to display in a view</td>
<td>Only allowed for sections in the view that match the current scope.</td>
<td>This restriction is independent of a user's role. Administrators cannot bypass this restriction.</td>
</tr>
<tr>
<td>Change the order of sections in a view</td>
<td>Only allowed for views that match the current scope.</td>
<td>This restriction is independent of a user's role. Administrators cannot bypass this restriction.</td>
</tr>
<tr>
<td>Select fields to display in a section</td>
<td>Only allowed for sections that match the current scope.</td>
<td>This restriction is independent of a user's role. Administrators cannot bypass this restriction.</td>
</tr>
<tr>
<td>Add or remove section columns</td>
<td>Only allowed for sections that match the current scope.</td>
<td>This restriction is independent of a user's role. Administrators cannot bypass this restriction.</td>
</tr>
<tr>
<td>Delete a form section</td>
<td>Only allowed for sections that match the current scope.</td>
<td>This restriction is independent of a user's role. Administrators cannot bypass this restriction.</td>
</tr>
<tr>
<td>Create new fields</td>
<td>Only allowed for sections that match the current scope and when the Allow configuration option is enabled.</td>
<td>This restriction is independent of a user's role. Administrators cannot bypass this restriction.</td>
</tr>
</tbody>
</table>

**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.

![Form Design](image)

**Figure 2: Visual indicators of editable sections**

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

• Section headings with a grey background.
• A grey 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.
Figure 3: Visual indicators of read-only sections

Default form design permissions

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

Table 3: Default 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</td>
<td>Allowed</td>
</tr>
<tr>
<td>application scope</td>
<td></td>
</tr>
<tr>
<td>Create new fields in sections belonging to the same</td>
<td>By default, denied. Requires application designer to set</td>
</tr>
<tr>
<td>application scope</td>
<td><strong>Allow configuration</strong> for application table.</td>
</tr>
<tr>
<td>Add or remove fields from sections belonging to the</td>
<td>Allowed</td>
</tr>
<tr>
<td>same application scope</td>
<td></td>
</tr>
<tr>
<td>Change the order of fields in sections belonging to</td>
<td>Allowed</td>
</tr>
<tr>
<td>the same application scope</td>
<td></td>
</tr>
<tr>
<td>Form design action</td>
<td>Permission setting</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Change the order of sections belonging to the same application scope</td>
<td>Allowed</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.
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 grey background.
- No buttons to add or remove fields.
- Save button with a grey 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

### Configuring Table Form

**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
- Room Resources
- New...

### Create New Field

**Name**

**Type**
- String

**Field length**
- Small (40)
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.

**Table 4: 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.
Table 5: 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</td>
</tr>
<tr>
<td></td>
<td>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 application tables permit other application scopes runtime access to:

- Read calls from the script API.
- All web service calls.

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 access settings.
5. 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.
6. 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 6: Configuration record creation restrictions

<table>
<thead>
<tr>
<th>Configuration record type</th>
<th>Creation restrictions when target table is in another application scope</th>
</tr>
</thead>
</table>
| Access Controls           | • You can only create field-level access controls with a role-based requirement.  
                            | • You cannot create table-level access controls for a table in another application scope.  
                            | • You cannot create field-level access controls that apply to all fields.  
                            | • You cannot create access controls that use conditions.  
                            | • You cannot create access controls that use a script-based condition. |
| 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. |
### Configuration record type

<table>
<thead>
<tr>
<th>Configuration record type</th>
<th>Creation restrictions when target table is in another application scope</th>
</tr>
</thead>
</table>
| 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.  |

### Parts of an application

An application has many possible parts. You must decide which parts are needed for your application.
Figure 6: Application model
Custom application record

The custom application record defines and identifies an application and all of 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.

A custom application record tracks all the configuration records associated to a particular application, and each configuration record is only assigned to one application at a time. When the application is ready for distribution, share it with other instances in the company or publishing it to the ServiceNow marketplace.

Application data tables

Data tables store application-specific data and provide a UI for users to access this data. Application developers create data tables and their associated lists and forms for application users to add and update records. An application owns its data tables and determines whether other applications can access resources from these tables. For example, the Book Rooms application might store conference room data in the Conference Rooms [x_acme_book_rooms_conference_rooms] table.

The system controls access to data tables from the user interface using standard access control rules. By default, the application creation process restricts access to application data tables by requiring users to have an application-specific user role. The system controls access to data tables from APIs and script calls using the application access settings for the table.

User interface elements

The system automatically creates a list and form view for each application data table. Application developers can select what fields are visible to users for each list and form view. They may also want to create custom menus, modules, or UI pages to allow users to find, add, and modify the information they need. For more information, see Administering Application Menus and Modules or UI Pages.

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 as needed to support the application data model but must manually create any associated access controls, menu role requirements, or module 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.

**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.
**Application File - Caller Close**

<table>
<thead>
<tr>
<th>Display name</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caller Close</td>
<td>Global</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Update name</th>
<th>Protection policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_script_28beab035f201000b12e3572f</td>
<td>-- None --</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Replace on upgrade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Rule</td>
<td>No</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’s translations.</td>
</tr>
</tbody>
</table>

**Application file form**

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

**Table 7: Application file form**

<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>Customer update</td>
<td>Flag that indicates whether the related record has customer changes. The value is automatically set to <strong>true</strong> when you change any field on the related 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>Replace on upgrade</td>
<td>Flag that indicates whether to overwrite customer changes when the plugin or application it belongs to is next upgraded. The value is <strong>false</strong> by default, preserving customer changes. To ensure the system overwrites customer changes during the next software upgrade, change the value to <strong>true</strong> on a local application file record. See Overwriting Customizations During Upgrades.</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>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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>

**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 make any changes without first 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:** Should you revert a customized file to its baseline state, then the record inherits the new protection policy as well. This could result in 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:** The Application File Types table provides system functionality and should not be modified.

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.
Application access settings

Application developers can use table application access settings to allow or prevent the creation of application files for their application tables.

You can specify what application artifacts are available to other custom applications in different application scopes. These permissions are in addition to the standard access controls (ACLs) that determine whether users can access data in the custom application table from the user interface.

Application developers can also protect script-based application files by making them read-only or hiding them from application users. For more information about scripting options, see Scripting in Scoped Applications.
Application access fields

The following fields control access to application data tables.

<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>
</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.

**Table 9: Runtime access points**

<table>
<thead>
<tr>
<th>Access points</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script API</td>
<td>Any supported object or method call from the scoped system API such as a GlideRecord call.</td>
</tr>
<tr>
<td>Web services</td>
<td>Any supported web service call such as a REST, JSON, or SOAP web service.</td>
</tr>
</tbody>
</table>

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.

**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.
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**.

**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 10: Default 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>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>

![Table - Marketing Event](image)

**Figure 8: Application access permissions for a table record**

**All runtime access restriction**

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.
Table 11: 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>

Figure 9: 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.
Figure 10: Deny all runtime access permissions to application tables

All runtime access permission
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.
Table 12: 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>

Figure 11: 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.
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
• UI actions
• Client scripts

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. For more information, see Additional Configuration Record Restrictions.

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.
   See the examples below for more information.
6. Click Update.

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 13: Default design-time 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>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>
Figure 13: Default access permissions to configuration records

*Example denying all design access*

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.

**Table 14: 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>
Figure 14: 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.
Figure 15: Limiting design access to this application scope only

Example granting access to configuration records
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.

Table 15: 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 is 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 is set to All application scopes</td>
</tr>
<tr>
<td>Dictionary entry (new field only)</td>
<td>• Can read is selected</td>
</tr>
<tr>
<td>UI actions</td>
<td>• Allow configuration is selected</td>
</tr>
</tbody>
</table>
The following diagram illustrates the effect of granting other application scopes the ability to create configuration records.

Figure 16: Granting other application scopes design access permission
Figure 17: Granting access to configuration records

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.
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.
4. Click **Submit**.

5. Test the fix script and make any necessary changes.

**Fix Script form**

Use the Fix Script form to update an application or data after installing or updating it.

<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>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Run once</td>
<td>Select the check box to run the script only one time (default). The fix script is ignored for subsequent upgrades.</td>
</tr>
<tr>
<td></td>
<td>Clear the check box allow the script to run every time the application is installed or upgraded.</td>
</tr>
<tr>
<td></td>
<td>This option is enforced by the following mechanism.</td>
</tr>
<tr>
<td></td>
<td>• When a fix script is processed, a corresponding record is added to the sys_update table.</td>
</tr>
<tr>
<td></td>
<td>• The upgrade process automatically skips fix scripts that are flagged as Run once and have an entry in the sys_update table.</td>
</tr>
<tr>
<td></td>
<td>Run once is not enforced when you test the fix script.</td>
</tr>
<tr>
<td>Flush cache</td>
<td>Select the check box to require a cache flush after the application is installed or upgraded.</td>
</tr>
<tr>
<td></td>
<td>Certain system changes—such as changes to the dictionary, client scripts, UI policy, or system properties—require a cache flush to take effect.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>If the check box is not selected for any of the fix scripts that are run, then the cache is not flushed.</td>
</tr>
<tr>
<td></td>
<td>Flush cache is enforced when you test the fix script.</td>
</tr>
<tr>
<td>Before</td>
<td>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).</td>
</tr>
<tr>
<td>Description</td>
<td>Enter a description of the fix script.</td>
</tr>
<tr>
<td>Script</td>
<td>Enter the code for the fix script.</td>
</tr>
</tbody>
</table>

**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>Proceed in Background</td>
<td>Use this option for long-running scripts, or if you do not know the expected execution time.</td>
</tr>
<tr>
<td>Proceed</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. From Related Links, click **Run Fix Script**.

**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 scope**

Each application has an application scope that determines which of its resources are available to other parts of the system.

Application scoping ensures that one application does not impact another application. You can specify what parts of the application other applications can access by setting application access settings.

For example, suppose 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 such as incident management cannot make changes to it without 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, an application developer or an API call can make to an application artifact. Application developers specify an application's scope when they create it.
Private application scope

Applications in a private application scope restrict access to their application artifacts so that only application artifacts in the same scope have full access to create, modify, remove, or run application data.

As the application developer, you set what parts of an application are accessible from other application scopes with application access settings.

By default, all new applications belong to their own application scope set to the application name, and they allow other applications to read their records with script calls. For example, a conference room booking application can have its own tables and business logic in the Conference Room Booking private application scope. While it can allow other applications to read its records such as a list of available conference rooms, it can prevent other applications from overwriting protected data such as reservation schedules.

Global scope

Applications in the global scope are like shared resources that any application developer can modify.

Global scope applications do not have a unique namespace identifier included in their application artifact names, but they can have their own application access permissions. Typically, only applications provided by ServiceNow are in the global scope, however all 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 ServiceNow marketplace. ServiceNow recommends creating all new applications in a private application scope. If you who want existing custom applications to take advantage of application scope, you must move the application from the global scope.

Application scope by application type

The provider and associated release version determine the application type.

<table>
<thead>
<tr>
<th>Table 16: Application types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application type</td>
</tr>
<tr>
<td>Baseline applications</td>
</tr>
<tr>
<td>Custom applications from prior versions</td>
</tr>
<tr>
<td>New custom applications</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 17: Application scope by application type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application type</td>
</tr>
<tr>
<td>Baseline applications</td>
</tr>
<tr>
<td>Custom applications from prior versions</td>
</tr>
</tbody>
</table>
Namespace identifier

The system adds the namespace identifier to the front of all application tables, fields, and script names to uniquely identify the application artifacts.

As the application developer, you cannot change or remove the identifier from application artifacts. This ensures that application artifacts are associated to the proper application and that all application artifacts have a unique name.

Namespace identifiers have this format:

\[x_\{Vendor\ prefix\}_\{Application\ ID\}\]

The system computes the application namespace identifier from the following elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Requirements</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The prefix character for custom applications</td>
<td>This string is always the x character.</td>
<td>x</td>
</tr>
<tr>
<td>The separator character for namespace identifier prefixes</td>
<td>This string is always the underscore character. The separator also appears between other elements in the prefix.</td>
<td>_</td>
</tr>
<tr>
<td>The instance vendor prefix (glide.appcreator.company.code)</td>
<td>This string is two to five characters long. ServiceNow generates this prefix for each customer. The instance stores the prefix in the glide.appcreator.company.code system property.</td>
<td>acme</td>
</tr>
<tr>
<td>The application ID</td>
<td>This string can be up to 40 characters long. Typically the ID matches the application name with spaces replaced by underscores.</td>
<td>book_rooms</td>
</tr>
</tbody>
</table>

The example values above generate a namespace identifier of x_acme_book_rooms.

**Note:** When you create a table or field, you cannot modify the namespace identifier but you can modify the rest of the name.

*Namespace identifier examples*

The following examples illustrate generating namespace identifiers for applications, tables, and fields.
### Table 19: 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 <strong>Capacity</strong> 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 <strong>Theme</strong> 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>
<tr>
<td>7. Select Marketing Events in the application picker and add the <strong>Theme</strong> field to the Marketing Event table.</td>
<td>u_theme</td>
<td>Custom fields in the global scope use the u_ prefix. To dot-walk to the field in a script, you would use u_marketing_event.u_theme.</td>
</tr>
</tbody>
</table>

**Note:** This example assumes that the Marketing Event table allows other application scopes to add fields. For more information, see Application Access Settings.
Table 20: Table updates

<table>
<thead>
<tr>
<th>Table name</th>
<th>New or updated</th>
<th>Description</th>
<th>Extends</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package [sys_package]</td>
<td>New</td>
<td>Contains a record for each baseline application activated from a plugin.</td>
<td>None</td>
</tr>
<tr>
<td>Application [sys_scope]</td>
<td>New</td>
<td>Contains a record for each application installed on the instance.</td>
<td>Package [sys_package]</td>
</tr>
<tr>
<td>Store Applications [sys_store_app]</td>
<td>New</td>
<td>Contains a record for each application downloaded from the ServiceNow App Store.</td>
<td>Application [sys_scope]</td>
</tr>
<tr>
<td>Custom Application [sys_app]</td>
<td>Updated</td>
<td>Contains a record for each custom-authored application on the instance.</td>
<td>Application [sys_scope]</td>
</tr>
</tbody>
</table>

Field updates for application scope

Table 21: Field updates

<table>
<thead>
<tr>
<th>Field</th>
<th>New or Updated</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>New</td>
<td>[Read only] The application scope that contains the record. A value of <strong>global</strong> means the record belongs to the global application scope. Any other value means the record belongs to a private application scope.</td>
</tr>
<tr>
<td>Table</td>
<td>Updated</td>
<td>By default, lists of tables only show tables and database views that are in the same application scope as the current record. You can show or hide a the table for a custom application by setting its application access controls.</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.

The application list displays the following information.
Table 22: Applications list

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developed</td>
<td>Displays applications application developers have created on this instance.</td>
</tr>
<tr>
<td>Downloaded</td>
<td>Displays applications administrators have downloaded on this instance.</td>
</tr>
<tr>
<td>Updates</td>
<td>Displays 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.
Figure 19: Application picker
Contextual development edit messages

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

Figure 20: 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.

Figure 21: 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 versioning

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 if there are updates available from the ServiceNow application repository or marketplace.
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 management

The system offers several ways to manage applications. You must have the admin role to perform these procedures.
Table 25: Application management actions

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

Install an application

Administrators can install applications developed by their own company from the application repository.

Install applications developed by your company from the application repository.

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.

If an application is not listed, contact your application developer to request they make the application available to other instances.

- You can only install one application at a time. The system disables the Install button for other applications during installation.
The installation progress appears.

Install an update

Administrators can install updates from the application repository to ensure they have the most recent application version.

Install updates from the application repository and ensure that the most recent application version is implemented.

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 it.

Select an application from the applications list

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.

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.

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.

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 as 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>None. The system clone process will copy this application version onto the target instance during the clone.</td>
</tr>
</tbody>
</table>

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

3. Log in to the clone target instance.

4. 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.</td>
</tr>
<tr>
<td></td>
<td>2. Load the update set containing the current application version.</td>
</tr>
<tr>
<td>The application was never installed on the source instance.</td>
<td>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 sub-production instance to address these requests. As development nears completion, you want to update your sub-production instance to the latest copy of production for some comprehensive testing.

You publish the Marketing Events version 2.0 application to an update set and retrieve the update set on your production instance. Version 2.0 of the Marketing Events application is now a local but uncommitted update set on the production instance.

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 delete version 1.0 of the Marketing Events application and then commit the local update set for version 2.0 of the application. The development instance now has version 2.0 of the Marketing Events application available for further development and testing.

ServiceNow applications

The system manages applications provided by ServiceNow separately from custom applications.

Applications provided by ServiceNow are in the global scope and cannot be downloaded or updated from the application repository or marketplace. Instead, the system tracks changes to these applications in the upgrade history.

Administrators can see the active and available applications provided by ServiceNow from the list of plugins.

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.
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>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</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.

```
Confirmation

This will delete the application 'Book Rooms'.

It will also delete all 20 files contained in this application. Show files.

[Cancel] [Delete]
```

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**

Administrators can uninstall applications they no longer need on an instance.

Uninstalling an application removes all application files associated with the application. You have the option to remove application data as well.

**Note:** You cannot uninstall applications on the **Develop** tab, but you can delete them.

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

3. Select the global application to which you want to move the records.

4. Click **Move**.

   The record and all of its descendant records are moved to the selected global application.
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 your application's files, create new files as you develop, and modify existing application files in a tabbed environment.

The system opens Studio whenever you edit a scoped application.
Figure 22: Studio

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 New interface.
- Navigate to files using familiar search-by-name or by-type behavior with the Go To dialog.
- Find code both within and outside their 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 their own instances or the ServiceNow Store.

**Note:** Studio is not intended for global applications and may behave unexpectedly if used to edit a global application.

### Table 28: Parts of the Studio UI

<table>
<thead>
<tr>
<th>UI element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td></td>
</tr>
<tr>
<td><strong>App Options Menu</strong></td>
<td>Contains a list of application-specific options.</td>
</tr>
<tr>
<td></td>
<td>• Create New Application file</td>
</tr>
<tr>
<td></td>
<td>• Publish</td>
</tr>
<tr>
<td></td>
<td>• Configure Application</td>
</tr>
<tr>
<td></td>
<td>• Switch Applications</td>
</tr>
<tr>
<td>Application name</td>
<td>The header displays the name of the currently-selected application.</td>
</tr>
<tr>
<td>Go To</td>
<td>Search for application files by name or type.</td>
</tr>
<tr>
<td>Code Search</td>
<td>Search within application files for a text string.</td>
</tr>
<tr>
<td></td>
<td>Search options include:</td>
</tr>
<tr>
<td></td>
<td>• Restrict search to a particular table</td>
</tr>
<tr>
<td></td>
<td>• Include all applications</td>
</tr>
<tr>
<td>Application Explorer</td>
<td></td>
</tr>
<tr>
<td>Collapse All</td>
<td>Collapses all nodes in the application explorer.</td>
</tr>
<tr>
<td>Expand All</td>
<td>Expand all nodes in the application explorer.</td>
</tr>
<tr>
<td>Data Model &gt; Tables</td>
<td>A list of application tables.</td>
</tr>
<tr>
<td></td>
<td>Click on a table name to display and edit it in the content frame.</td>
</tr>
<tr>
<td>Access Control</td>
<td>A list of application access elements such as:</td>
</tr>
<tr>
<td></td>
<td>• Roles</td>
</tr>
<tr>
<td></td>
<td>• Access Controls</td>
</tr>
<tr>
<td></td>
<td>Click on a record name to display and edit it in the content frame.</td>
</tr>
<tr>
<td>UI element</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **Navigation** | A list of application navigational elements such as:  
  • Application Menus  
  • Modules  
  • Application Menus (Mobile)  
  • Modules (Mobile)  
  Click on a record name to display and edit it in the content frame. |
| **Content frame** | |
| **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 on a tab to display and edit the record.  
  A tab with a blue circle icon indicates the record contains unsaved changes. |

## Access Studio

Studio allows developers to add and update application files.

You must create an application to open it with Studio.

Role required: admin

1. Navigate to **System Applications > Studio**.
Welcome to Studio

ServiceNow Studio allows you to rapidly build applications that help your business run better.

![Welcome to Studio](image)

The system displays the Welcome to Studio page.

2. To view an existing applications, click **Go** from **View Applications**.
The system displays a list of applications.

3. To create a new application, click **Go** from **Create Application**.
The system displays a list of application creation options.

ServiceNow studio keyboard shortcuts

ServiceNow Studio supports a variety of keyboard shortcuts to manage and edit application files.

<table>
<thead>
<tr>
<th>Keyboard combination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows: Control+Shift+O</td>
<td><strong>Go To.</strong> Open any file in your application.</td>
</tr>
<tr>
<td>Mac: Command+Shift+O</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 New.
   You can also use a 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>
<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>
</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 <strong>Filter</strong> 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**

Studio allows application developers to search for application files by name and by type.

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>Table 31: Studio keyboard shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyboard combination</strong></td>
</tr>
<tr>
<td>Windows: Control+Shift+O</td>
</tr>
<tr>
<td>Mac: Command+Shift+O</td>
</tr>
</tbody>
</table>

Studio opens the Go To pop-up window.

4. Enter a search string.

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.

<table>
<thead>
<tr>
<th>Table 32: Studio keyboard shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyboard combination</strong></td>
</tr>
<tr>
<td>Windows: Control+Shift+F</td>
</tr>
<tr>
<td>Mac: Command+Shift+F</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 contains search of the application files you selected. While the search is running the 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. Search results are grouped 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.

Table 33: 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 Scope, Menu, or User role 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>
<tr>
<td>Note:</td>
<td>To publish the application in the ServiceNow Store, the version must conform to the application certification standards.</td>
</tr>
<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 Application scope on page 48.</td>
</tr>
<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>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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 this application is tracked by subscription management.</td>
</tr>
<tr>
<td>Subscription requirement</td>
<td>Specifies whether this application requires a separate subscription.</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 <a href="#">Create an application menu</a>.</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 <a href="#">Create a role</a>.</td>
</tr>
<tr>
<td>Short description</td>
<td>Enter a description of the application purpose or usage.</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> on page 43.</td>
</tr>
</tbody>
</table>
Switch between an application

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 you from making some changes when the application files you want to edit belong to another application. Switching to the application that owns the application files ensures that you make changes in the proper application.

1. Navigate to **App Options Menu > Switch Applications**.
   The system displays the instance's list of applications.
2. Click the application you want to switch to.
   The system reloads Studio to display the selected application.

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>Sharing method</th>
<th>Makes available to</th>
<th>Typical use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish internally to the company’s instances</td>
<td>All instances assigned to the same company</td>
<td>Transfer the latest application version to a test or production environment.</td>
</tr>
<tr>
<td>Publish to the ServiceNow Store</td>
<td>All ServiceNow customers</td>
<td>Share or sell applications to other companies.</td>
</tr>
<tr>
<td>Publish to an Update Set</td>
<td>Any instance with access to the Update Set file</td>
<td>Save a version of an application for compliance or backup reasons.</td>
</tr>
<tr>
<td>Push to team development instances</td>
<td>Other instances in the team development environment</td>
<td>Push developer changes to the parent instance.</td>
</tr>
</tbody>
</table>

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.
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 the application repository**

Uploading an application to the application repository makes this version of the application available to all instances assigned to the same company.
Installing an application from the application repository ensures that you always have the same application version on all instances, that the instance meets the application installation requirements, and allows you to install application updates.

1. Navigate to **System Applications > Applications**.
2. Click the **Develop** tab.
3. Open the application record you want to upload to the application repository.
4. Click the **Make App available on other instances** related link.
5. Optional: Fill in the fields, as appropriate (see table)

### Make App available on other instances

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>[Read only] The company associated with this instance.</td>
</tr>
<tr>
<td>Vendor Prefix</td>
<td>[Read only] The unique vendor prefix associated with your company.</td>
</tr>
<tr>
<td>App Name</td>
<td>[Read only] The application name as specified in the custom application record.</td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Dev Notes</td>
<td></td>
</tr>
</tbody>
</table>
6. Click **Submit**.
   The system uploads the current version of the application to the application repository store allowing other instances to download it.

*When to use the application repository*

The application repository enables you to deploy completed applications to end users. Depending on your organization needs, you can use different deployment options.

<table>
<thead>
<tr>
<th>Deployment option</th>
<th>Good for</th>
<th>Future considerations</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

*Make App Available fields*

Use the Make App available on other instances form to share applications with the application repository.

**Table 36: Make App Available fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>[Read only] The company associated with this instance.</td>
</tr>
<tr>
<td>Vendor Prefix</td>
<td>[Read only] The unique vendor prefix associated with your company.</td>
</tr>
<tr>
<td>App Name</td>
<td>[Read only] The application name as specified in the custom application record.</td>
</tr>
<tr>
<td>Version</td>
<td>The application version uploaded.</td>
</tr>
<tr>
<td>Dev Notes</td>
<td>[Optional] A description of the changes contained in this release.</td>
</tr>
</tbody>
</table>
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 you must:

• 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.

Publish an application to an Update Set

Publishing an application creates an Update Set containing the current version of the application's 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 > 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><img src="image" alt="Version field" /></td>
</tr>
<tr>
<td>Description</td>
<td><img src="image" alt="Description field" /></td>
</tr>
<tr>
<td>Include demo data</td>
<td><img src="image" alt="Include demo data" /></td>
</tr>
</tbody>
</table>

![Publish button](image)

![Cancel button](image)

---

**Table 37: Publish to Update Set**

© 2017 ServiceNow. All rights reserved.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 <strong>Version</strong> field. The Update Set has the name <code>&lt;Application name&gt; - &lt;Version&gt;</code>. If you leave this field blank, the initial Update Set has the name <code>&lt;Application name&gt;</code> and subsequent Update Sets have the name <code>&lt;Application name&gt; - &lt;Sequential number&gt;</code>.</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>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. <strong>Warning:</strong> 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 <a href="#">Import sets</a>. <strong>Note:</strong> • 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 counter in the sample data or the counter on the target instance. • For translated fields, only records in English are transferred.</td>
</tr>
</tbody>
</table>

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. For more information, see Transferring Update Sets.

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

*Publish to Update Set fields*

Use the Publish to Update Set form to associate an application with an update set.

### Table 38: Publish to Update Set fields

<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 Application name - Version. If you leave this field blank, the initial Update Set has the name Application name and subsequent update sets have the name Application name - Sequential number.</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>

**Caution:** 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. Administrators can adjust the maximum number of data records to include with an application.

**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 counter in the sample data or the counter on the target instance.
- For translated fields, only records in English are transferred.

**Limit data included**

Administrators can limit the number of data records to include when publishing an application to an Update Set.

This limit only applies to application data. An Update Set always contains all of the configuration records associated with the application at the time of its creation.

1. Enter `sys_properties.list` in the navigation filter.
2. Locate the `com.snc.apps.publish.maxrows` property.
3. Set the **Value** field.
   - To minimize performance impact, set this value to less than or equal to the default value of 1,000.
4. Click **Update**.

**Automatic recovery of draft records**

Studio can maintain a version of any open existing record with unsaved changes. This allows users to recover unsaved changes should their user session end 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 if 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](image)

**Figure 23: Automatic recovery message**

For each recovered record, users have the option to:
• 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 the developer studio handles the recovery of draft records by navigating to **Auto Recovery > Properties.**

| **Table 39: Auto recovery properties** |

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Auto Recovery</td>
<td>Allows users to recover unsaved changes while working in the Studio.</td>
</tr>
<tr>
<td>glide.ui.auto.recovery</td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Comma separated list of tables not supported for auto recovery</td>
<td>glide.ui.auto.recovery.unsupported.tables</td>
</tr>
<tr>
<td></td>
<td>The comma-separated list of tables excluded from automatic recovery.</td>
</tr>
<tr>
<td>Comma separated list of field types not supported for auto recovery</td>
<td>glide.ui.auto.recovery.unsupported.field.types</td>
</tr>
<tr>
<td></td>
<td>The comma-separated list of field types excluded from automatic recovery.</td>
</tr>
<tr>
<td>Comma separated list of field types to exclude from auto recovery</td>
<td>glide.ui.auto.recovery.exclude.field.types</td>
</tr>
<tr>
<td></td>
<td>The comma-separated list of field types you want to exclude from automatic recovery.</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 40: Auto recovery properties**

<table>
<thead>
<tr>
<th>Dictionary attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exclude Auto Recovery</td>
<td>glide.ui.auto.recovery.unsupported.tables</td>
</tr>
<tr>
<td>exclude_auto_recovery</td>
<td>Disables automatic recovery of draft records for this table.</td>
</tr>
</tbody>
</table>

**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**

**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.
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 Fulfillers**

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**

An administrator can activate the Service Creator plugin to access the application.

1. Navigate to **System Definition > Plugins**.
2. Right-click the plugin name on the list and select **Activate/Upgrade**.
   - If the plugin depends on other plugins, these plugins are listed along with their activation status.
3. 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 policy when you first activate the plugin on a development or test instance. You can load demo data after the plugin is activated by repeating this process and selecting the check box.
4. Click **Activate**.

**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

Table 41: 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</td>
<td>Tracks fulfillers for a service category. Use these records to grant or remove remove roles as needed.</td>
</tr>
<tr>
<td>[catalog_category_request_user]</td>
<td></td>
</tr>
<tr>
<td>Service [sc_cat_item_producer_service]</td>
<td>Stores all services.</td>
</tr>
<tr>
<td>Service Category App Menu</td>
<td>Stores the application menus for each service category.</td>
</tr>
<tr>
<td>[service_category_app_menu]</td>
<td></td>
</tr>
<tr>
<td>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>
<tr>
<td>[service_category_user_role]</td>
<td></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>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>UI policy</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</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>
<tbody>
<tr>
<td>glide.citizen_developer.category.auto_publish</td>
<td>Automatically add new service categories to the service catalog as subcategories of the Departmental Services category.</td>
</tr>
<tr>
<td></td>
<td>• Type: true</td>
</tr>
<tr>
<td></td>
<td>• Default value: true</td>
</tr>
<tr>
<td></td>
<td>• Location: System Properties [sys_properties] table</td>
</tr>
<tr>
<td>glide.citizen_developer.set_category_roles</td>
<td>Comma-separated list of roles that can set the category for a new service.</td>
</tr>
<tr>
<td></td>
<td>• Type: String</td>
</tr>
<tr>
<td></td>
<td>• Default value: admin,catalog_admin</td>
</tr>
<tr>
<td></td>
<td>• Location: System Properties [sys_properties] table</td>
</tr>
<tr>
<td>glide.service_creator.auto_add_to_category</td>
<td>Automatically add new services to the Departmental Services service catalog category, in addition to the department-specific category.</td>
</tr>
<tr>
<td></td>
<td>• Type: true</td>
</tr>
<tr>
<td></td>
<td>• Default value: true</td>
</tr>
<tr>
<td></td>
<td>• Location: System Properties [sys_properties] table</td>
</tr>
</tbody>
</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>
</tbody>
</table>
### Script include

<table>
<thead>
<tr>
<th>Script include</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 when appropriate.</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>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>Business rule</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</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**

<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>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------</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:

#### Table 43: Tables Created with New Service Categories

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Department Name&gt;</code> Tasks <code>[&lt;service category table name&gt;]</code></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 <strong>Table name</strong> 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 <code>[sys_numbers]</code> record.</td>
</tr>
</tbody>
</table>

#### Table 44: 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 <strong>Table name</strong> 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>
### Table 45: 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 worknote.</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 worknote.</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>

### Table 46: Forms Created with New Service Categories

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;department name&gt;</code> 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>

### Table 47: Service Catalog Categories

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;service category name&gt;</code></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.

Table 48: Roles

<table>
<thead>
<tr>
<th>Role Title [Name]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Service Category&gt; User [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 administrator approves the request to publish the service category. When the category is published, the service category manager can assign service category editors and service fulfillers, and create services to offer in the service 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**.
Table 49: 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>
<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>
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.
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.
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**.

### 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.

### 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.

---

**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.

---

**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 Userservice 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**.

---

**Add a question**

You can add a question using the service workspace.

1. Use one of these methods to add a question:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Types tab</td>
<td>Click the <strong>Types</strong> tab and select the cell for the question type you want to add.</td>
</tr>
<tr>
<td>Use the new question of the same type button (+)</td>
<td>Click the add a new question of the same type button (+) on an existing question.</td>
</tr>
</tbody>
</table>

2. Drag the question to the location where it should appear on the catalog item.
3. Click the question to edit the question properties.
4. Enter the question text and other relevant values.
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.

Service design interface

The service creator includes a service design interface that service category managers and editors use to create and edit services. When the service is complete, you can publish it to create a catalog item that end users can request.

The service design interface appears in a new browser tab when a service category manager or editor creates a new service or edits an existing service. When working with a service, first access the published service category for that service by navigating to Service Creator > My Service Categories.

The interface contains several sections with distinct purposes.
Figure 24: Service design layout
Service description fields

The fields in this area are used for all services. The information you enter describes the service, rather than any individual service request. Information entered in these fields appears on the catalog item for the service.

Table 50: Service description fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the service, which becomes the catalog item name.</td>
</tr>
<tr>
<td>Request Title</td>
<td>A one-line description of the item that becomes a subtitle for the catalog item.</td>
</tr>
<tr>
<td>Image</td>
<td>The image to display when a user views the associated catalog item in the desktop interface. This field appears when the service is configured with Desktop and Mobile or Desktop Only availability. Allowed file types depend on your instance configuration.</td>
</tr>
<tr>
<td>Description</td>
<td>A description of the service. This description appears when a user views the item in the service catalog.</td>
</tr>
</tbody>
</table>

Service workspace

The service workspace is the primary area for working with questions. The service workspace allows you to add questions and arrange them in an appropriate sequence for the service catalog item.

The question's location in the workspace represents its relative position on the catalog item. Each question has an identifying label and a handle icon for selecting and moving it.

The service workspace can be divided into sections. Sections organize questions into groups. All services must have at least one section. A choice list on the section header allows you to specify if the questions in a section should appear in one or two columns. You can add or remove a section by clicking the add a new section icon (+) or the delete this section icon (X) on an existing section.

Question and settings navigator

The left side of the service design interface provides several tabs allowing you to add questions, set question properties, and configure fulfillment and mobile settings for the service.
Table 51: Tabs

<table>
<thead>
<tr>
<th>Tab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question Types</td>
<td>The Types tab provides a list of question types you can add to the service. These may include generic types that you can use to create custom questions, or preconfigured, commonly-used questions defined by an administrator. Each preconfigured question may be added to a particular service only once. Generic types may be added multiple times. For example, a tuition reimbursement service might ask the requesting user for the name of the course using a generic Single Line Text question, or for the employee that completed the course using a preconfigured reference to the User [sys_user] table.</td>
</tr>
<tr>
<td>Question Properties</td>
<td>The Properties tab allows you to configure the details of a question, such as the Label that appears as the question text, and default values. The properties displayed depend on the question type. For example, for a multiple choice question you can define the question text, the list of available choices, and the default choice.</td>
</tr>
<tr>
<td>Fulfillment</td>
<td>Allows you to configure catalog availability for the service and default assignment rules for service requests.</td>
</tr>
<tr>
<td>Mobile</td>
<td>Allows you to configure the availability and appearance of the catalog item for mobile devices.</td>
</tr>
</tbody>
</table>

Table 52: Fulfillment settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Category</td>
<td>The service catalog category this service should appear under. A service may appear in more than one service catalog category. Selecting a category causes a new Target Category field to appear. The service catalog categories the service appears in do not necessarily match the name of the service category.</td>
</tr>
<tr>
<td>Target Table</td>
<td>[Read-only] The table this service creates records on, as determined by the service category the service belongs to.</td>
</tr>
<tr>
<td>Fulfillment Group</td>
<td>The group assigned to complete tasks associated with this service. The assignment can be changed manually for individual tasks as needed.</td>
</tr>
</tbody>
</table>
Table 53: Mobile settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fulfillment User</td>
<td>The user assigned to complete tasks associated with this service. The assignment can be changed manually for individual tasks as needed.</td>
</tr>
<tr>
<td>Availability</td>
<td>The platforms where this service can be requested. Choose from Desktop and Mobile, Desktop Only, or Mobile Only.</td>
</tr>
<tr>
<td>Mobile Image</td>
<td>The image to display when a user views the associated catalog item in the mobile interface. This field appears when the Availability setting is Desktop and Mobile or Mobile Only. Allowed file types depend on your instance configuration.</td>
</tr>
</tbody>
</table>

Create or edit an unpublished service

The service category manager or editor can create and edit services within that category.

1. Navigate to Service Creator > My Service Categories.
2. Open a service category with a State of Published to Catalog.
3. On the Service Category form, open the service design interface using one of these methods:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related link</td>
<td>Click the Create New Service related link.</td>
</tr>
<tr>
<td>Related list</td>
<td>Right-click a service in the Draft Services related list and select Edit.</td>
</tr>
</tbody>
</table>

4. Configure any service settings, and add or modify questions.

   Note: For more information on questions, see Add a question on page 100 and Create a preconfigured question on page 106.

Edit a published service

The service category manager or editor can edit services within that category.

When editing a published service, ServiceNow resets the State of the service record from Published to Draft. Changes to the service must be published before the changes appear in the service catalog. The original service is still available until the new draft record is published. Publishing that draft record replaces the original published service.

1. Navigate to Service Creator > My Published Services.
2. Right-click a service and select Edit.
3. Configure any service settings, and add or modify questions.
4. Click **Save.**

**Questions**

Questions added to a service are presented to users who request the service through the service catalog. The answers entered by the requesting user provide the details for that specific service request.

Add, remove, and configure questions and sections using the service workspace.

You can define your own questions of a specific type, or use a **preconfigured question.**

**Create a preconfigured question**

Preconfigured questions provide a quick way to add commonly-used questions to a service. References to other records can only be used as preconfigured questions, not as generic question types. Each preconfigured question may be added to a particular service only once.

An administrator can create preconfigured questions using the catalog item designer.

1. Navigate to **Item Designer > Question Types.**
2. Click **New.**
3. Select the **Preconfigured question** check box.
4. Enter question details.
5. Click **Submit.**

**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.**

**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.

**Getting started with Update Sets**

Before working with Update Sets, create a standard process for moving customizations from instance to instance. When working with Update Sets, be sure to avoid common pitfalls.
For example, an update set called Incident Management 2.0 might hold a set of enhancements to incident management. While Incident Management 2.0 is marked as the current update set, all changes are tracked in it.

An update set consists of:

- 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.

By default, update sets only track changes to baseline applications and platform features. This allows developers to create functionality on a sub-production instance and promote the changes to another instance.

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 contain.
- 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.

Application developers have 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.

When to use Update Sets

Update Sets are useful for storing and applying changes. This topic gives an overview of when to use Update Sets and when to use other deployment options.

<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. Storing and applying a particular version of an application. Producing a file for export.</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>
</tbody>
</table>

**Note:** Do not use Update Sets to install applications. Instead, use the application repository or the ServiceNow Store to install applications.
<table>
<thead>
<tr>
<th>Deployment option</th>
<th>Good for</th>
<th>Future considerations</th>
</tr>
</thead>
<tbody>
<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>Works best when each development team has 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 sub-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>

**Plan the update process for an Update Sets**

Because update sets make changes to an instance, review the following best practice information to avoid errors and performance issues. Learn how to plan the update process to ensure that there are no problems during the process.

Before working with Update Sets, create a standard process for moving customizations from instance to instance.

1. Check that both instances are 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 sub-production instances.
- Clone the production instance onto the sub-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. Best practice is to move Update Sets from dev to test and then from test to production.

5. Plan for when to commit the Update Set to production. Avoid committing an Update Set to a production instance during business hours. The instance may perform slower temporarily as the Update Set 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. Know how Update Sets work.
   - What records are generated for an Update Set
   - Which customizations are tracked by Update Sets
   - Which dictionary changes are valid for Update Sets
   - Which customizations can be backed out (reversed) once applied

8. Before making any customizations, double-check that the correct Update Set is selected.

Avoid pitfalls when working with an Update Set

Because Update Sets make changes to an instance, review the following best practice 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 Sets 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 Sets 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.

Customizations in an Update Set

An Update Set is an XML file that contains a list of changes to an instance. Administrators can save an Update Set as a local file that can be transferred to another instance.

Typically you create 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

UI actions for exporting and importing customizations in an XML file

• Export to XML on the table sys_update_set
• Export to XML on the table sys_remote_update_set
• Import Update Set from XML on the table sys_remote_update_set

The Export to XML UI action on sys_update_set 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 ExportWithRelatedLists Script Include, which exports a record and manually defined related lists to a single XML file.

Select an Update Set

Update Sets track changes as you develop. Follow this procedure to specify the update set that should collect the updates that you make:

1. Open the Settings panel.
2. Select the desired Update Set from the Update Set picker. Only Update Sets that are appropriate for the current application scope appear in the selection list. The associated application is displayed in square brackets.
Now, any customization that you perform on a tracked table is recorded by the Update Set. For example:
• An update record is added or updated in the current Update Set.
• A new version record is created with the current Update Set as the source.

You can compare versions of any customized object, and you can revert changes to an older version.

Save an Update Set as an 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 a customization from a single XML file

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. Follow standard procedure to commit the Update Set.

Update Set transfers to other instances

When an Update Set is completed, you can transfer it to another instance 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. Add a property using sys_properties.list.

System Update Sets

An Update Set is a group of customizations that can be moved from one instance to another. Update sets allow administrators to group a series of changes into a named set and then move them as a unit to other systems. In most cases, Update Sets allow customizations to be developed in a development instance, moved to a test instance, and then applied to a production 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. Add a property using sys_properties.list.

Before using Update Sets, review the Getting Started with Update Sets page.

Update sets 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.

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's 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.

How the system determines the default Update Set

Only one Update Set can be the default set for any 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 Update Set

The *global default Update Set* is the set where Default set = true and application scope is global.
The global default Update 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.

Global default update 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 update set (regardless of the Name of the set) provides system functionality and should not be changed, deleted, or moved between systems.

When the system auto-generates a default Update 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.

Here are some common cases where the system auto-generates a default Update Set:

When the user logs in for the first time

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 the default set is marked as Ignored or Completed

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.

When the user changes application scope

The system auto-generates a new default Update Set for a scope when the following conditions occur:

• You change application scope and
• Your preferred Update Set is Complete or Ignored and
• There is no In-Progress default Update Set for the new scope
Update Sets for changes to applications

The system creates a separate Update Set for each application that only contains changes associated with the application. This ensures that each application’s access settings are properly evaluated and applied when committing Update Set changes.

Customizations in an Update Set

An Update Set is an XML file that contains a list of changes to an instance. Administrators can save an Update Set as a local file that can be transferred to another instance.

Typically you create 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

Tracked customizations

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 specifically 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, the order requests, items, and catalog tasks are not tracked by update sets.

Update sets have a limited capacity to transfer data as application files. This is intended to provide demo data for applications. For larger data transfers export data and import it with an import set or web service.

`Update_synch` attribute

A default rule blocks the use of the `update_synch` attribute on a table for which it is not predefined to avoid issues.

To see the list of tables where customizations are tracked, navigate to System Definition > Dictionary and filter on attributes CONTAINS update_synch.

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.

**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.
Special handlers

Some changes require special handlers because they represent information on multiple tables.

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 and system update sets

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.

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, suppose you create a new Activity [u_activity] table that extends the Task table and 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. In addition, you have the option to move the u_activity table and its associated choice to a separate application without affecting the default ServiceNow task application.

Dictionary changes and update sets

Update sets prevent you from applying dictionary changes that result in data loss.

Blocked dictionary changes include:

- Removing tables
- Changing a column's 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.

Homepages and content pages in Update Sets

Homepages and content pages are not added to Update Sets by default.

Homepages and content pages are not added to Update Sets by default. You must manually add pages to the current Update Set by unloading them.

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.
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.

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.

To exclude a field from updates, add the `update_exempt` dictionary attribute to the field.

**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 re-activate 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.

Any changes that are only the result of changing the **Active** field are excluded from update tracking. 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`

Overwrite a customization 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**. Alternatively, right-click the header and select **Show Application File**.
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.

Grant access to the Update Set Picker

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.

**Role required:** admin

1. **Add the system property** `glide.ui.update_set_picker.role` to the System Properties table.
2. Set the value of glide.ui.update_set_picker.role to the role you want to give access to the update set picker.

**Note:**

This property accepts only one role. To give access to multiple roles, create a new role (for example, update_set_picker) and include that role in the roles that need access.

---

**Track an application table in Update Sets**

Application developers can have the system automatically track application tables in Update Sets. Application developers may want to track application changes in an Update Set to save or distribute a particular version of an application.

1. During table creation, set **Extends Table** to Application File [sys_metadata].
2. Continue table creation to add columns and dictionary settings.

**Create an Update Set**

An Update Set is a group of customizations that can be moved from one instance to another.

For example, an Update Set called Incident Management 2.0 might hold a set of enhancements to incident management. While Incident Management 2.0 is marked as the current Update Set, all changes are tracked in it.

1. Navigate to **System Update Sets > Local Update Sets** and click **New**.
2. Enter Update Set details as described 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:
   1. Create the Update Set.
   2. Select the new Update Set in the Update Set picker.
   3. Add customizations to the new Update Set.
### Table 54: 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>State</td>
<td>Select <strong>In progress</strong> for a new Update Set. Selecting an In progress Update Set tracks customizations in the Update Set. The <strong>Update Set picker</strong> only displays In progress Update Sets. Select <strong>Completed</strong> only when you are certain that the Update Set is complete. After it is marked Completed, do not set it back to In progress. Instead, create a new Update Set with further customizations, and make sure to commit the Update Sets in the order that they were marked Completed. Use Completed 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 Completed Update Sets on the production instance to Ignore. This state ensures the Update Set is not committed again when cloning the instance.</td>
</tr>
</tbody>
</table>
Select an Update Set

Update Sets track changes as you develop. Follow this procedure to specify the update set that should collect the updates that you make:

1. Open the Settings panel.
2. Select the desired Update Set from the Update Set picker. Only Update Sets that are appropriate for the current application scope appear in the selection list. The associated application is displayed in square brackets.
Now, any customization that you perform on a tracked table is recorded by the Update Set. For example:
• An update record is added or updated in the current Update Set.
• A new version record is created with the current Update Set as the source.

You can compare versions of any customized object, and you can revert changes to an older version.

View a change in an Update Set

View the customizations that make up an Update Set and to compare the update to the current version.

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**.
Update Set - Incident Release 1

- Name: Incident Release 1
- Application: Incident.App
- Created: 44 minutes ago
- Created by: admin
- Merged to: 

Description: Tailoring Incident for a new group that is coming on board

Related Links
Merge With Another Update Set

Customer Updates (14)
- Created
- View
- Target name
- Updated by

- about a minute ago
  - Type: Table
  - Incident Table 1
  - Updated by: admin

- about a minute ago
  - Type: Dictionary
  - Incident Table 1
  - Updated by: admin
Customer update records

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

Compare a local Update Set

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.

   ![Progress](attachment:progress.png)

   - Go to the Collision Report when the report is complete.
   - The Update Set Collisions list appears, showing all the changes in the selected sets.
4. Inspect the list for collisions by locating duplicate Collision Numbers that show the same change in separate Update Sets.
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 records or customized objects

You can navigate between a customer update record and the customized object or the application file for the object.

- To navigate from an update record to the customized object, such as the application menu record: Click the **Show Related Record** related link.
- To navigate from an update record to the application file record for the object: Click the **Show Application File** related link.
Figure 25: Show Record

- 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**.
View a report on customizations and configuration changes

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.

How collisions occur and how to avoid them

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_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:

```plaintext
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:

```plaintext
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. The resulting identical record name in each instance helps the system to identify collisions even if the records have different Sys IDs.

```plaintext
sys_dictionary
sys_documentation
sys_choice_set
```
sys_ui_list
sys_ui_related_list

To prevent creating duplicate records with Update Sets:

- 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 .
- Enable a unique index for the table from the system dictionary. See .

**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 as 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.

## 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 55: New Remote Instance

<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>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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></td>
<td><strong>Note:</strong> You cannot change the URL after the system verifies the connection. Use the Active field to deactivate unwanted remote instances.</td>
</tr>
<tr>
<td>Username</td>
<td>Enter the user on the remote instance who authorizes transferring Update Sets to this 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 Sets, navigate to **System Update Sets > Retrieved Update Sets**.
Transfer an update set 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 certain steps before transferring an update set.

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 the following steps before transferring an update set:

1. Contact customer support to find out the IP addresses of all application nodes supporting your instance.
3. Add the IP address from step one as an exception.

Transfer an update set with basic authentication

If the source instance has basic authentication turned on for SOAP requests, you must use valid credentials to retrieve update sets.

If the source instance has basic authentication turned on for SOAP requests, you must use valid credentials to retrieve update sets.

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.

2. Click Preview Update Set.
   
   The Update Set Preview page shows results and lists problems. Read the information and click Close.
3. 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.

<table>
<thead>
<tr>
<th>Problems addressed: 0 Ignored</th>
<th>0 Skipped</th>
<th>5 Remaining</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Vehicle Update Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Global</td>
</tr>
<tr>
<td>Update source</td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>Previewed</td>
</tr>
<tr>
<td>Loaded</td>
<td>2015-02-03 13:11:38</td>
</tr>
</tbody>
</table>

**Related Links**

Show All Preview Records

**Update Set Preview Problems (5)**

<table>
<thead>
<tr>
<th>Update Set Preview Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go to Type</td>
</tr>
</tbody>
</table>

- **Update Set = Vehicle Update Set > Status =**

- **Could not find a table field (x_snc_test_vehicle.status) referenced in this update**

- **Find missing field**
- **Find missing update**
- **Accept remote update**
- **Skip remote update**
4. Open the Update Set record and click **Show All Preview Records** to make sure the correct updates are being committed.

5. 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.

Resolve a preview problem with an Update Set

The process of previewing an Update Set detects problems that may occur if you commit the updates on the local instance. After you preview and before you commit an Update Set, follow this procedure to resolve all of the problems that the preview process discovered.

2. Open the Update Set record and scroll down to the Update Set Preview Problems related list.
3. Review each problem description to determine the cause and resolve the problem as described in the table. Alternatively, choose one of the following options:
   - **Accept remote update**: Commit the remote Update Set without fixing the problem.
   - **Skip remote update**: Skip the update when you commit the update set.

<table>
<thead>
<tr>
<th>Type and Example</th>
<th>Description</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing object</td>
<td>The object or a referenced object does not exist on the target instance. For example:</td>
<td>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 the following Available Actions to assist in problem resolution:</td>
</tr>
<tr>
<td></td>
<td>• An update modifies the form layout for a table that has not been created in the local instance.</td>
<td>• <strong>Find missing field</strong> or <strong>Find missing record</strong>: Opens a new window and searches the source instance for the missing field (dictionary entry) or record.</td>
</tr>
<tr>
<td></td>
<td>• A UI policy action is included in the update set, but the parent UI policy is not.</td>
<td>• <strong>Find missing update</strong>: Opens a new window and searches the source instance for the update record that corresponds to the missing field or record.</td>
</tr>
<tr>
<td>Type and Example</td>
<td>Description</td>
<td>Resolution</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Collision              | A change in the Update Set is older than a change to the same object on the local instance. | 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     | The object exists in another remote Update Set that has not been committed. | 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 | 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.) | 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

<table>
<thead>
<tr>
<th><strong>Type and Example</strong></th>
<th><strong>Description</strong></th>
<th><strong>Resolution</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Application scope validation issue</td>
<td>The previewer identifies the following combination of states as a problem:</td>
<td>Transfer the Update Set only to instances with that include the application scope or ensure that the update set includes the application.</td>
</tr>
<tr>
<td></td>
<td>• The scope for the Update Set is not Global and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The application is not found on the target instance and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The application is not included with the update set and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The application is not found on the ServiceNow Store.</td>
<td></td>
</tr>
</tbody>
</table>

### 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.
**Table 57: Update Set Preview**

<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.

**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.

1. Navigate to **System Update Sets > Retrieved Update Sets** and open the Update Set.
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**.

   A completion page appears when the Update Set has been successfully committed.

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 will 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 to be 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 Sets on the production instance, you should always change the state to **Ignore**. This state ensures 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.

**Note:** Do not back out the **Default** Update Set because that can damage the configuration of the instance.

**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.

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. See **Expected results of the back out process for an update set** on page 140

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. You must resolve each problem before proceeding with the back out.
   
   • To keep the latest version, click **Use Current**.
   • To back out to the previous version, click **Back Out**.

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.
Expected results of the back out process for an update set

The back out process reverses both record-level updates and changes to the dictionary.

**Warning:** Some changes that result from a back out result in data loss.

<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>

Delete an update set

Deleting an Update Set is strongly not recommended. 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 may choose to delete the original sets. This function is restricted by an access control rule (ACL) on the Update Set `sys_update_set` table.

**Warning:** Deleting Update Sets that contain `sys_update_xml` entries is restricted and strongly not recommended because:

- Deleting the Update Set does not undo the updates.
- Deleting the Update Set removes any record of who applied the customizations.
- Deleting the Update Set removes the `sys_update_xml` entries associated with the Update Set, so customizations are overwritten when the instance is upgraded.

To undo customizations, back out the Update Set rather than deleting it.

Delete an update entry

Deleting `sys_update_xml` entries is not recommended.

**Warning:** ServiceNow strongly recommends that you **do not delete** `sys_update_xml` entries because:

- Deleting a `sys_update_xml` entry removes the record of modifications to the instance.
• Your customizations may be overwritten when the instance is upgraded.

    When you try to delete an update entry, a warning message appears. Click OK to confirm the deletion.

Merge Update Sets

You can merge multiple update sets into a single update set to simplify the transfer process. 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.
Figure 27: Merge update sets

3. Enter a **Name** for the new update set. Updates are added to this set when the original sets are merged.

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.

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.

Team Development

Team Development supports parallel development on multiple, sub-production ServiceNow instances.

Team Development provides the following:

- 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.</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></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></td>
</tr>
<tr>
<td></td>
<td></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>Works best when each development team has 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 sub-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>
Team Development process

1. Set up the development 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.

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.
Table 59: 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 Show Changes</td>
<td>To view a comparison between the current local version and the version since Last Pull. An error message appears if a previous version does not exist (for example, in the case of a newly created record). If a previous version is available, you can revert to that version from the comparison window</td>
</tr>
<tr>
<td>Right-click the row and select Show Application</td>
<td>To open the application file for the customized record.</td>
</tr>
<tr>
<td>File</td>
<td></td>
</tr>
<tr>
<td>Right-click the row and select Show Version</td>
<td>To open the current version record.</td>
</tr>
</tbody>
</table>

Pull exceptions

Pulling ignores versions when certain conditions occur.

Table 60: 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 &quot;previously resolved collision&quot;.</td>
</tr>
<tr>
<td>Skipped</td>
<td>Pulls will skip versions where there is a problem with the version record such as a corrupt or missing version.</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* on page 164.
• **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* on page 164.
• **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.
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:
   • 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 that they submitted and 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.

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.

   ![Change current parent instance](image)
   
   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 have been approved and applied to the parent instance.
   • Pushes in the Collided stage have been rejected because of a collision.
   • Pushes in the Awaiting Code Review stage are awaiting review.
   • Pushes in the Code Changes Rejected stage have been rejected by a reviewer.
   • Pushes in the Code Review Request Cancelled stage have been cancelled 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>Remote Instance</th>
<th>8381</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Push</td>
<td>Latest version date</td>
<td>2014-01-24 11:15:17</td>
</tr>
<tr>
<td>Created</td>
<td>2014-01-24 11:15:17</td>
<td>Stage</td>
<td>Compile</td>
</tr>
<tr>
<td>Created by</td>
<td>admin</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please review my changes

### Related Links

Team Dashboard

<table>
<thead>
<tr>
<th>Push and Pull Versions</th>
<th>Reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Push or Pull = Test Push 001

<table>
<thead>
<tr>
<th>Approver Name</th>
<th>Approver ID</th>
<th>Decision</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Administrator</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 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. 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.

1. Ensure 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.

<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>Resolve a collision by taking the parent version</td>
<td>There is no longer a local change to ignore</td>
</tr>
<tr>
<td>Action</td>
<td>Result</td>
</tr>
<tr>
<td>--------</td>
<td>--------</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.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>To stop ignoring changes</strong></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><strong>To stop ignoring changes and add them to the queue instead</strong></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><strong>To remove changes that you do not want to push</strong></td>
<td>Select the check boxes beside the rows and select <strong>Do Not Push</strong> from the Actions choice list</td>
</tr>
<tr>
<td><strong>To add changes</strong></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

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

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><strong>To remove changes from the queue</strong></td>
<td>Select the check boxes beside the rows and select <strong>Do Not Push</strong> from the Actions choice list.</td>
</tr>
<tr>
<td><strong>To remove changes from the queue and choose to ignore them instead</strong></td>
<td>Select the check boxes beside the rows and select <strong>Ignore This Change</strong> from the Actions 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. 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>
<tr>
<td>To load the version pulled from the parent as the current version</td>
<td>Click Use Pulled Version.</td>
</tr>
</tbody>
</table>
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 on page 166 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]
• short_field_name [Short Field Name]
• short_table_name [Short Table Name]
• slushbucket [Slush Bucket]
• source_id [Source ID]
• source_name [Source Name]
• source_table [Source Table]
• string_boolean [<none>]
• sys_class_name [System Class Name]
• sysrule_field_name [System Rule Field Name]
• table []
• text []
• time []
• timer [Timer]
• translated [Translated]
• tree_code [Tree Code]
• tree_path [Tree Path]
• user_image [Image]
• user_input [User Input]
• variables [Variables]
• version [Version]
• video [Video]
• week_of_month [Week of Month]
• wide_text [Wide Text]
• wms_job [WMS Job]
• workflow [Workflow]

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 sub-production instances, administrators can set up the Team Development instance hierarchy and grant access rights for developers.
Access rights for developers

There must be a set of credentials for each instance in the Team Development hierarchy in order to use Team Development.

To use Team Development, developers must have a set of credentials for each instance in the team development hierarchy.

Table 62: 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 register a remote instance</td>
<td>One of following:</td>
</tr>
<tr>
<td>Right to push changes to the parent instance from a</td>
<td>• A user with the admin role on the parent instance</td>
</tr>
<tr>
<td>Right to compare to a registered remote instance</td>
<td>• A user with the admin role on the registered development instance</td>
</tr>
<tr>
<td>Access to the 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_user role on the parent instance</td>
</tr>
</tbody>
</table>

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

1. Navigate to Team Development > Exclusion Policy.
2. Click New.
3. Complete the Exclusion Policy form (see table).
4. Click Submit.

Table 63: 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>
</tbody>
</table>
| Policy           | Select when the policy applies. Options include:  
|                  | • Push only  
|                  | • Push and Pull  
|                  | • Pull only  
| Remote Instance  | [Optional] Select a specific remote instance to ignore changes from during pull operations. Leaving this field blank ignores changes from all remote instances. |
| Table            | Select which table to ignore changes for. |
| Conditions       | Select any additional criteria a change must meet to be ignored other than the table name. This field is only visible when the Policy is **Push only**. |

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 steps 1–5 for each instance in the hierarchy that this instance needs to push and pull with (for example, Sub-Dev 2 and Sub-Dev 3).

<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>Field</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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>
<tr>
<td>Short description</td>
<td>[Optional] Enter any other relevant information about the remote instance.</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.

---

**Caution**: 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 65: Team dashboard control panel options

<table>
<thead>
<tr>
<th>Option</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 steps 1–2 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.

Caution: 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. 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 parent instance’s remote instance record.

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>Table 66: Code review notification table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notification name</td>
</tr>
</tbody>
</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 cancelled review | Push or Pull [sys_sync_history] | • The user who cancelled review  
• The push that was cancelled |

Code review workflow

The team Development Code Review workflow manages how changes are pushed to the parent.
By default this workflow:

- Starts when changes are pushed to the parent instance.
- Verifies 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.

**Figure 28: Team development code review workflow**

⚠️ **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 sub-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 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 or lists, right-click the header and select **Configure > Form Layout** or **Configure > 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 be 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.

\[
\begin{array}{|c|c|}
\hline
\text{Field} & \text{Description} \\
\hline
\text{Name} & \text{A unique identifier for coalescing versions of the same customized record.} \\
\hline
\text{Record name} & \text{Name of the customized record.} \\
\hline
\text{Source} & \text{Indicator of how the version was added on the instance.} \\
& \begin{itemize}
\item System Upgrade: from a software upgrade (the baseline version).
\item Update Set: from an update set that was created or committed on the instance.
\item Pull History: from a pull in Team Development.
\end{itemize} \\
\hline
\text{State} & \text{Indicator of whether the version is or has ever been loaded on the instance.} \\
& \begin{itemize}
\item Current: the version is currently loaded.
\item Previous: the version has previously been loaded on the instance. When a current version is replaced by a new version, it becomes a previous version.
\item History: the version was never loaded on the instance and was only inserted for historical purposes, such as when pulling versions from the parent in Team Development.
\end{itemize} \\
\hline
\text{Application} & \text{The application for the customized record, if it is assigned to an application.} \\
\hline
\text{Payload} & \text{The data for this version of the customized record.} \\
\hline
\text{Additional fields on the list view} & \\
\hline
\end{array}
\]

© 2017 ServiceNow. All rights reserved. 181
### 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.

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 <strong>Compare to Current</strong>.</td>
</tr>
<tr>
<td>From the Update Versions form</td>
<td>Click the <strong>Compare to Current</strong> 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:
<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
</table>
| To resolve the differences by choosing the previous version            | • Team Development: Click **Revert to Selected Version**.  
• For a version of an object: Click **Use Local Version** to maintain the local record as the current version. The pulled version is added to the version history for the record. |
| To resolve the differences by modifying the current version and saving the merged changes | 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 > (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:  
• Team Development: Click **Save Merge** to save the changes to the current version.  
• Team Development: Click **Use Pulled Version** or **Use Local Version** option to accept or reject all changes, as appropriate.  
• On the upgrade history Compare to Current form, the only option is **Revert to Base System** |

### 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 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 will result 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 cannot 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.
Figure 30: 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.
Index

A
access to Update Set picker 117
active field tracking 117
administering Update Sets 116
application changes Update Sets 115
application development 60
application files adding 69
searching by name 71
by record values 71
by type 71
application repository best practice 79
applications a scope 49
access 20
access settings design time denying 39
granting 41
runtime default permissions 32
example denying 33
granting 35
fields 31
adding logic 20
application data tables user interface elements 24
user roles 24
application files form 28
protecting 29
relationships 29
viewing 25
application scope configure layout 12
creating lists and forms 9, 9
field updates 52
form design 9, 11
namespace identifier 50, 50
table updates 51
building 8
creation options create a custom application 6
start from a template 6
start from an existing service 6, 7
start from global 7
start from scratch 5
custom application record 24
data tables 8
default 61
default view 16
deleting 61
development environment 64
development process 4
fix scripts running 47
testing 46
global scope 49
include data 76
install 57
install updates 58
limit data included 85
managing 56
moving 63
parts of 22
private scope 49
publish 81
publishing 77
related lists 16
removing 61
repository 77, 79
selecting 58, 58
sharing 76
studio 67
tracking changes in update sets 118
uninstall 62
update set 81
Update Set 84
updating 74
user interface 8
Applications module 3

B
backout 139
baseline version 181

C
code review Team Development 175
collisions limitations while resolving 166
resolving 164
Compare to Current 182
conflict resolving 164
contextual development environment application list 52
application picker 53
application version 55
context 48
resource throttling 56, 56
warning messages 55
CreateNow studio automatic recovery 86
draft records 86
custom application development 3

© 2017 ServiceNow. All rights reserved. 188
customizations
load from XML 112

D
default
update set, global 114
default Update Set 113, 114
delete
sys_update_xml entry 140
Update Sets 140
dependencies 43

E
excluding fields
Update Sets 117
export customizations
single XML file 110

G
global default
update set 114
grant access
Update Set picker 117

I
import customizations
single XML file 110

M
Managing Services 95, 97, 98, 99, 100, 101

N
new service categories
components 93
e-mail notifications 93
forms 93
service catalog categories 93
tables 93
user roles 93

O
overwriting
customizations
upgrade 117

P
parent instance 172
preserve applications 60
preview 136
preview problems
resolving 134

pushes and pulls
Team Development 178

R
resolving
collisions 164, 166

S
script includes
protection policy 56
Service Creator
installed with 88
service design interface
add a question 100
create a preconfigured question 106
create or edit an unpublished service 105
description fields 103edit a published service 105
publish a service 106
question and settings navigator 103
questions 106
service workspace 103
ServiceNow studio
automatic recovery 87
draft records 87
keyboard shortcuts 68
ServiceNow Studio
automatic recovery 85
draft records 85
share applications
applications 80
publish 80
store 80
sys_update_xml entry
delete 140
system clone 60
system Update Sets
system Update Sets 106, 113
tracking application changes 118
System Update Sets
choice lists 116
content pages 116
determine which customizations are tracked 115
homepages 116
special handlers 116
system update sets 113, 116
system Update Sets 115, 116, 116
transfer update sets 132, 132
update sets tables 113

T
tables 24
Team Development
access rights 169
approve or reject push 148
back out 149, 161
best practice 143
change the parent instance 151
code review 175
code review request 150
Compare to Current 182
compare to peer instances 154
calendar concept 176, 176, 177, 178, 185
dashboard 146
enable code review 172
exclusion policy 169
ignore 155
instance hierarchy 173
local changes 145, 145, 149, 155, 161, 161
local version 154
parent instance 172
process 145
pull a version 157
pull exceptions 146
push 161
push version 158
pushed change 151
pushed version 154
pushes and pulls 178
reconcile 163
remote instance 170
resolve collisions 164
resolve multiple collisions 168
review status 151
roles 178
setting up 168, 173
using 143
versions 178, 179, 180, 181, 184, 184
tracking
active field 117
transfer customizations
Update Sets 110, 115
Transfer Update Sets
basic authentication 132
ip access control 132

U
Update Set
default 113, 114
delete 140
load from XML 112
preview 132
save as XML 112
Update Set picker
granting access 117
update sets
dictionary changes 116
global default 114
merging 141
table changes 116
Update Sets
administering 116
avoid pitfalls 109
back out 139
best practice 107
commit 138
excluding fields 117
getting started 106
planning 108
retrieving 130

V
versions
Compare to Current 182
revert a change 184
suppress 184
transferring 180
viewing for an object 179
Versions
version records 181

W
Web services integrations 24