Package and Distribute Your Apps

Salesforce, Spring '21

'21

@salesforcedocs

Last updated: January 7, 2021

CONTENTS

PACKAGE AND DISTRIBUTE YOUR APPS

Want to distribute open-source projects to provide developers with the basic building blocks for an application? Or distribute and sell applications to customers? Packages let you do all that and more.

Find additional information in the *ISVforce Guide* or visit the Salesforce Partner Portal.

IN THIS SECTION:

Work with Packages

Create managed and unmanaged packages.

IN THIS SECTION:

Understanding Packages

A *package* is a container for something as small as an individual component or as large as a set of related apps. After creating a package, you can distribute it to other Salesforce users and organizations, including those outside your company.

Packages come in two forms—unmanaged and managed:

Unmanaged packages

Unmanaged packages are typically used to distribute open-source projects or application templates to provide developers with the basic building blocks for an application. Once the components are installed from an unmanaged package, the components can be edited in the organization they are installed in. The developer who created and uploaded the unmanaged package has no control over the installed components, and can't change or upgrade them. Unmanaged packages should not be used to migrate components from a sandbox to production organization. Instead, use Change Sets.

As a best practice, install an unmanaged package only if the org used to upload the package still exists. If that org is deleted, you may not be able to install the unmanaged package.

Managed packages

Note: Salesforce has two ways that you can build managed packages, first-generation packaging (1GP) and second-generation packaging (2GP). This guide describes 1GP. For new solutions, use 2GP as described in the Second-Generation Managed Packages section of the Salesforce DX Developer Guide.

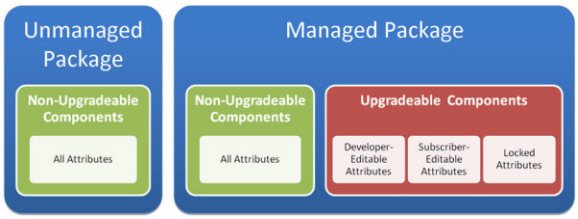
Managed packages are typically used by Salesforce partners to distribute and sell applications to customers. These packages must be created from a Developer Edition organization. Using the AppExchange and the License Management Application (LMA), developers can sell and manage user-based licenses to the app. Managed packages are also fully upgradeable. To ensure seamless upgrades, certain destructive changes, like removing objects or fields, can not be performed.

Managed packages also offer the following benefits:

- Intellectual property protection for Apex
- Built-in versioning support for API accessible components
- The ability to branch and patch a previous version
- The ability to seamlessly push patch updates to subscribers
- Unique naming of all components to ensure conflict-free installs

The following definitions illustrate these concepts:

Unmanaged and Managed Packages



EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create packages:
- Create AppExchange Packages

To upload packages to the AppExchange:
- Upload AppExchange Packages

Components

A *component* is one constituent part of a package. It defines an item, such as a custom object or a custom field. You can combine components in a package to produce powerful features or applications. In an unmanaged package, components are not upgradeable. In a managed package, some components can be upgraded while others can't.

Attributes

An *attribute* is a field on a component, such as the name of an email template or the `Allow Reports` checkbox on a custom object. On a non-upgradeable component in either an unmanaged or managed package, attributes are editable by both the developer (the one who created the package) and the subscriber (the one who installed the package). On an upgradeable component in a managed package, some attributes can be edited by the developer, some can be edited by the subscriber, and some are locked, meaning they can't be edited by either the developer or subscriber.

For information on which components can be included in a package and which attributes are editable for each component, see the *ISVforce Guide*.

Packages consist of one or more Salesforce components, which, in turn, consist of one or more attributes. Components and their attributes behave differently in managed and unmanaged packages.

If you plan to distribute an app, it is important to consider packaging throughout the development process. For example:

- While creating your app, consider how components and their attributes behave in different packages and Salesforce editions.
- While preparing your app for distribution, consider how you want to release it to your customers.
- While installing a package, consider your organization's security and license agreements.

SEE ALSO:

Manage Packages

Prepare Your Apps for Distribution

Glossary

The following terms and definitions describe key application and packaging concepts and capabilities:

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

Available in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

App

Short for "application." A collection of components such as tabs, reports, dashboards, and Visualforce pages that address a specific business need. Salesforce provides standard apps such as Sales and Service. You can customize the standard apps to match the way you work. In addition, you can package an app and upload it to the AppExchange along with related components such as custom fields, custom tabs, and custom objects. Then, you can make the app available to other Salesforce users from the AppExchange.

AppExchange

The AppExchange is a sharing interface from Salesforce that allows you to browse and share apps and services for the Lightning Platform.

Beta, Managed Package

In the context of managed packages, a beta managed package is an early version of a managed package distributed to a sampling of your intended audience to test it.

Deploy

To move functionality from an inactive state to active. For example, when developing new features in the Salesforce user interface, you must select the "Deployed" option to make the functionality visible to other users.

The process by which an application or other functionality is moved from development to production.

To move metadata components from a local file system to a Salesforce organization.

For installed apps, deployment makes any custom objects in the app available to users in your organization. Before a custom object is deployed, it is only available to administrators and any users with the "Customize Application" permission.

License Management Application (LMA)
A free AppExchange app that allows you to track sales leads and accounts for every user who downloads your managed package (app) from the AppExchange.

License Management Organization (LMO)
The Salesforce organization that you use to track all the Salesforce users who install your package. A license management organization must have the License Management Application (LMA) installed. It automatically receives notification every time your package is installed or uninstalled so that you can easily notify users of upgrades. You can specify any Enterprise, Unlimited, Performance, or Developer Edition organization as your license management organization. For more information, go to http://www.salesforce.com/docs/en/lma/index.htm.

Major Release
A significant release of a package. During these releases, the major and minor numbers of a package version increase to any chosen value.

Managed Package
A collection of application components that is posted as a unit on the AppExchange and associated with a namespace and possibly a License Management Organization. To support upgrades, a package must be managed. An organization can create a single managed package that can be downloaded and installed by many different organizations. Managed packages differ from unmanaged packages by having some locked components, allowing the managed package to be upgraded later. Unmanaged packages do not include locked components and cannot be upgraded. In addition, managed packages obfuscate certain components (like Apex) on subscribing organizations to protect the intellectual property of the developer.

Managed Package Extension
Any package, component, or set of components that adds to the functionality of a managed package. You cannot install an extension before installing its managed package.

Namespace Prefix
In a packaging context, a namespace prefix is a one to 15-character alphanumeric identifier that distinguishes your package and its contents from packages of other developers on AppExchange. Namespace prefixes are case-insensitive. For example, ABC and abc are not recognized as unique. Your namespace prefix must be globally unique across all Salesforce organizations. It keeps your managed package under your control exclusively.

Package
A group of Lightning Platform components and applications that are made available to other organizations through the AppExchange. You use packages to bundle an app along with any related components so that you can upload them to AppExchange together.

Package Dependency
This is created when one component references another component, permission, or preference that is required for the component to be valid. Components can include but are not limited to:

- Standard or custom fields
- Standard or custom objects
- Visualforce pages
- Apex code

Permissions and preferences can include but are not limited to:

- Divisions
- Multicurrency
- Record types

Package Installation
Installation incorporates the contents of a package into your Salesforce organization. A package on the AppExchange can include an app, a component, or a combination of the two. After you install a package, you may need to deploy components in the package to make it generally available to the users in your organization.

Package Version
A package version is a number that identifies the set of components uploaded in a package. The version number has the format `majorNumber.minorNumber.patchNumber` (for example, 2.1.3). The major and minor numbers increase to a chosen value during every major release. The `patchNumber` is generated and updated only for a patch release.

Unmanaged packages are not upgradeable, so each package version is simply a set of components for distribution. A package version has more significance for managed packages. Packages can exhibit different behavior for different versions. Publishers can use package versions to evolve the components in their managed packages gracefully by releasing subsequent package versions without breaking existing customer integrations using the package. See also Patch and Patch Development Organization.

Patch
A patch enables a developer to change the functionality of existing components in a managed package, while ensuring subscribing organizations that there are no visible behavior changes to the package. For example, you can add new variables or change the body of an Apex class, but you may not add, deprecate, or remove any of its methods. Patches are tracked by a `patchNumber` appended to every package version. See also Patch Development Organization and Package Version.

Patch Development Organization
The organization where patch versions are developed, maintained, and uploaded. Patch development organizations are created automatically for a developer organization when they request to create a patch. See also Patch and Package Version.

Patch Release
A minor upgrade to a managed package. During these releases, the patch number of a package version increments.

Publisher
The publisher of an AppExchange listing is the Salesforce user or organization that published the listing.

Push Upgrade
A method of delivering updates that sends upgrades of an installed managed package to all organizations that have installed the package.

Subscriber
The subscriber of a package is a Salesforce user with an installed package in their Salesforce organization.

Test Drive
A test drive is a fully functional Salesforce organization that contains an app and any sample records added by the publisher for a particular package. It allows users on AppExchange to experience an app as a read-only user using a familiar Salesforce interface.

Unmanaged Package
A package that cannot be upgraded or controlled by its developer.

Upgrading
Upgrading a package is the process of installing a newer version. Salesforce supports upgrades for managed packages that are not beta.

Uploading
Uploading a package in Salesforce provides an installation URL so other users can install it. Uploading also makes your packaged available to be published on AppExchange.

SEE ALSO:

Build Your Own Salesforce App

Creating Managed Packages

Creating a managed package is just as easy as creating an unmanaged package. The only requirement to create a managed package is that you're using a Developer Edition organization.

Before creating a managed package:

- Review the information about managed packages in the ISVforce Guide to determine if this is the right type of package for your app.
- Optionally, install the License Management Application (LMA) from http://sites.force.com/appexchange. Search for `License Management App` to locate it. The License Management Application (LMA) tracks information about each user who installs your app. It allows you to track what users have which version, giving you a means of distributing information about upgrades.

  The License Management Application (LMA) can be installed in any Salesforce organization except a Personal, Group, or Professional Edition organization and does not need to be the same Salesforce organization that you use to create or upload the package, although it can be. You can also use the same License Management Application (LMA) to manage an unlimited number of your managed packages in different Developer Edition organizations.
- Configure your developer settings. Your developer settings specify your namespace prefix, the Salesforce organization where you install the License Management Application (LMA), and the unmanaged package you want to convert into a managed package.

IN THIS SECTION:

SEE ALSO:

Manage Packages

Convert Unmanaged Packages to Managed

Configure Your Developer Settings

Register a Namespace Prefix

Specifying a License Management Organization

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To enable managed packages:

- Customize Application

To create packages:

- Create AppExchange packages

To upload packages:

- Download AppExchange packages

About Managed Packages

Note: Salesforce has two ways that you can build managed packages, first-generation packaging (1GP) and second-generation packaging (2GP). This guide describes 1GP. For new solutions, use 2GP as described in the Second-Generation Managed Packages section of the Salesforce DX Developer Guide.

A managed package is a collection of application components that are posted as a unit on AppExchange, and are associated with a namespace and a License Management Organization.

- You must use a Developer Edition organization to create and work with a managed package.
- Managed packages are depicted by the following icons:
  - Managed - Beta
  - Managed - Released
  - Managed - Installed

Tip: To prevent naming conflicts, Salesforce recommends using managed packages for all packages that contain Apex to ensure that all Apex objects contain your namespace prefix. For example, if an Apex class is called `MyHelloWorld` and your org's namespace is `OneTruCode`, the class is referenced as `OneTruCode.MyHelloWorld`.

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure Your Developer Settings

The developer settings in a Developer Edition organization allow you to create a single managed package, upload that package to the AppExchange, allowing other users to install and upgrade the package in their organization. After configuring your developer settings the first time, you can no longer modify them. Regardless of the developer settings, you can always create an unlimited number of unmanaged packages.

To configure your developer settings:

1. From Setup, enter *Packages* in the Quick Find box, then select **Packages**.
2. Click **Edit**.

   Note: This button doesn't appear if you've already configured your developer settings.

3. Review the selections necessary to configure developer settings, and click **Continue**.
4. Register a namespace prefix.
5. Choose the package you want to convert to a managed package. If you do not yet have a package to convert, leave this selection blank and update it later.
6. Click **Review My Selections**.
7. Click **Save**.

Tip: You may want to specify a License Management Organization (LMO) for your managed package; to find out more, go to http://sites.force.com/appexchange/publisherHome.

SEE ALSO:

Creating Managed Packages
Register a Namespace Prefix
Specifying a License Management Organization

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To configure developer settings:
- Customize Application

To create packages:
- Create AppExchange Packages

To upload packages:
- Upload AppExchange Packages

Register a Namespace Prefix

In a packaging context, a namespace prefix is a one to 15-character alphanumeric identifier that distinguishes your package and its contents from packages of other developers on AppExchange. Namespace prefixes are case-insensitive. For example, ABC and abc are not recognized as unique. Your namespace prefix must be globally unique across all Salesforce organizations. It keeps your managed package under your control exclusively.

Important: When creating a namespace, use something that's useful and informative to users. However, don't name a namespace after a person (for example, by using a person's name, nickname, or private information.)

Salesforce automatically prepends your namespace prefix, followed by two underscores ("__"), to all unique component names in your Salesforce organization. A unique package component is one that requires a name that no other component has within Salesforce, such as custom objects, custom fields, custom links, s-controls, and validation rules. For example, if your namespace prefix is abc and your managed package contains a custom object with the API name, Expense__c, use the API name abc__Expense__c to access this object using the API. The namespace prefix is displayed on all component detail pages.

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Warning: S-controls stored in the s-control library or the Documents tab that do not use the Lightning Platform API still function properly after you register a namespace prefix. However, s-controls stored outside of your organization or s-controls that use the Lightning Platform API to call Salesforce may require some fine-tuning. For more information, see S-control in the *Object Reference*.

Your namespace prefix must:

- Begin with a letter
- Contain one to 15 alphanumeric characters
- Not contain two consecutive underscores

For example, `myNp123` and `my_np` are valid namespaces, but `123Company` and `my__np` aren't.

To register a namespace prefix:

1. From Setup, enter *Packages* in the Quick Find box and select **Package Manager** or **Packages**, depending on your Setup menu.
2. In the Developer Settings panel, click **Edit**.

   Note: This button doesn't appear if you've already configured your developer settings.

3. Review the selections that are required for configuring developer settings, and then click **Continue**.
4. Enter the namespace prefix you want to register.
5. Click **Check Availability** to determine if the namespace prefix is already in use.
6. If the namespace prefix that you entered isn't available, repeat the previous two steps.
7. Click **Review My Selections**.
8. Click **Save**.

SEE ALSO:

Creating Managed Packages
Configure Your Developer Settings
Specifying a License Management Organization

Specifying a License Management Organization

A license management organization is a Salesforce organization that you use to track all Salesforce users who install your managed package. The license management organization receives notification (in the form of a lead record) when a user installs or uninstalls your package and tracks each package upload on Salesforce AppExchange.

Your license management organization can be any Salesforce Enterprise, Unlimited, Performance, or Developer Edition organization that has installed the free License Management Application (LMA) from AppExchange. To specify a License Management Organization, go to http://sites.force.com/appexchange/publisherHome.

SEE ALSO:

Creating Managed Packages
Configure Your Developer Settings
Register a Namespace Prefix

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Manage Packages

A package is a collection of Lightning Platform components and applications that are made available to other organizations through the AppExchange. A managed package is a collection of application components that are posted as a unit on AppExchange, and are associated with a namespace and a License Management Organization. To support upgrades, a package must be managed. An organization can create a single managed package that can be downloaded and installed by many different organizations. Managed packages differ from unmanaged packages by having some locked components, allowing the managed package to be upgraded later. Unmanaged packages do not include locked components and cannot be upgraded. In addition, managed packages obfuscate certain components (like Apex) on subscribing organizations to protect the intellectual property of the developer.

For details, see the *ISVforce Guide*.

To manage your packages, enter *Packages* in the Quick Find box, then select **Packages**. For more customization, see Configure Your Developer Settings on page 8.

From the list of packages, you can:

- Click **New** to create a new package, enter a package name and description, and click **Save** to store it in your list of packages.
- Click **Edit** to update the package properties.
- Click **Del** to delete the package. The components contained in your package are not deleted.
- Click the name of the package to view the details of the package.

Note: To create a test drive or choose a License Management Organization (LMO) for what you have uploaded, click **Proceed to AppExchange** from the package upload detail page.

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To upload packages:
- Upload AppExchange packages

IN THIS SECTION:

Permission Sets and Profile Settings in Packages

Developers can use permission sets or profile settings to grant permissions and other access settings to a package. When deciding whether to use permission sets, profile settings, or a combination of both, consider the similarities and differences.

SEE ALSO:

Understanding Packages

View Package Details

Create a Package

About Package Versions

A package version is a number that identifies the set of components uploaded in a package. The version number has the format `majorNumber.minorNumber.patchNumber` (for example, 2.1.3). The major and minor numbers increase to a chosen value during every major release. The `patchNumber` is generated and updated only for a patch release. Unmanaged packages are not upgradeable, so each package version is simply a set of components for distribution. A package version has more significance for managed packages. Packages can exhibit different behavior for different versions. Publishers can use package versions to evolve the components in their managed packages gracefully by releasing subsequent package versions without breaking existing customer integrations using the package.

Version numbers depend on the package release type, which identifies the way packages are distributed. There are two kinds:

Major Release

A major release denotes a Managed - Released package. During these releases, the major and minor numbers of a package version increase to a chosen value.

Patch Release

A patch release is only for patch versions of a package. During these releases, the patch number of a package version increments.

The following table shows a sequence of version numbers for a series of uploads:

Upload Sequence	Type	Version Number	Notes
First upload	Managed - Beta	1.0	The first Managed - Beta upload.
Second upload	Managed - Released	1.0	A Managed - Released upload. Note that the version number does not change.
Third upload	Managed - Released	1.1	Note the change of the minor release number for this Managed - Released upload. If you are uploading a new patch version, you can't change the patch number.
Fourth upload	Managed - Beta	2.0	The first Managed - Beta upload for version number 2.0. Note the major version number update.
Fifth upload	Managed - Released	2.0	A Managed - Released upload. Note that the version number does not change.

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

When an existing subscriber installs a new package version, there is still only one instance of each component in the package, but the components can emulate older versions. For example, a subscriber may be using a managed package that contains an Apex class. If the publisher decides to deprecate a method in the Apex class and release a new package version, the subscriber still sees only one instance of the Apex class after installing the new version. However, this Apex class can still emulate the previous version for any code that references the deprecated method in the older version.

Package developers can use conditional logic in Apex classes and triggers to exhibit different behavior for different versions. This allows the package developer to continue to support existing behavior in classes and triggers in previous package versions while continuing to evolve the code.

When you are developing client applications using the API, you can specify the version of each package that you use in your integrations.

SEE ALSO:

- Manage Packages
- Planning the Release of Managed Packages
- *Apex Developer Guide*
- *SOAP API Developer Guide*

Create a Package

Packages are containers for distributing custom functionality between Salesforce orgs. Create a package to upload your app or Lightning component to the AppExchange or to deploy changes between orgs.

Tip: Before you begin, determine if you want to create and upload a managed or unmanaged package.

1. From Setup, enter *Packages* in the Quick Find box, then select **Packages**.
2. Click **New**.
3. Enter a name for your package. You can use a different name than what appears on AppExchange.
4. From the dropdown menu, select the default language of all component labels in the package.
5. Optionally, choose a custom link from the Configure Custom Link field to display configuration information to installers of your app. You can select a predefined custom link to a URL or s-control that you have created for your home page layouts; see the Configure Option on page 75. The custom link displays as a **Configure** link within Salesforce on the Salesforce AppExchange Downloads page and app detail page of the installer's organization.
6. Optionally, in the Notify on Apex Error field, enter the username of the person to notify if an uncaught exception occurs in the Apex code. If you do not specify a username, all uncaught exceptions generate an email notification that is sent to Salesforce. This option is only available for managed packages. For more information, see *Handling Apex Exceptions in Managed Packages*.

   Note: Apex can only be packaged from Developer, Enterprise, Unlimited, and Performance Edition organizations.

7. Optionally, in the Notify on Packaging Error field, enter the email address of the person who receives an email notification if an error occurs when a subscriber's attempt to install, upgrade, or uninstall a packaged app fails. This field appears only if packaging error notifications are enabled. To enable notifications, contact your Salesforce representative.

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create packages:
- Create AppExchange Packages

8. Optionally, enter a description that describes the package. You can change this description before you upload it to AppExchange.
9. Optionally, specify a post install script. You can run an Apex script in the subscriber organization after the package is installed or upgraded. For more information, see Running Apex on Package Install/Upgrade.
10. Optionally, specify an uninstall script. You can run an Apex script in the subscriber organization after the package is uninstalled. For more information, see Running Apex on Package Uninstall.
11. Click **Save**.

SEE ALSO:

View Package Details

Prepare Your Apps for Distribution

Add Components to Your Package

After you have created a package, you need to add components to it, such as app, object, Apex classes or Visualforce pages. These packages can be uploaded to share with others privately or posted on Salesforce AppExchange to share publicly.

To add components to a package, from Setup, enter *Packages* in the Quick Find box, then select **Packages**. Next, select the name of the package that you want to add components to. From the package detail page:

1. Click **Add Components**.
2. From the dropdown list, choose the type of component you want to add to your package.
   - At the top of the list, click a letter to display the contents of the sorted column that begin with that character.
   - If available, click the **Next Page** (or **Previous Page**) link to go to the next or previous set of components.
   - If available, click **fewer** or **more** at the bottom of the list to view a shorter or longer display list.
3. Select the components you want to add.

   Note: Some components cannot be added to Managed - Released packages. For a list of these components, see *Developing Packages for Distribution*.

   S-controls cannot be added to packages with restricted API access.
4. Click **Add To Package**.
5. Repeat these steps until you have added all the components you want in your package.

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create packages:
- Create AppExchange Packages

SEE ALSO:

Manage Packages

View Package Details

Prepare Your Apps for Distribution

Components Available in Managed Packages

A component is part of a managed package, such as custom objects and custom fields. You can combine components in a package to produce powerful features or applications. In managed packages, you can upgrade some components.

Not all components can be packaged for distribution. If you create an app with components that aren't packageable, your subscribers must create and configure the components after they install your app. Keep packageable components in mind as you develop.

The following components are available in a managed package.

Upgradeable
Some components are updated to a newer version when a package is upgraded.

- **No**: The component isn't upgraded.
- **Yes**: The component is upgraded.

Subscriber Deletable
A subscriber or installer of a package can delete the component.

- **No**: The subscriber can't delete the component.
- **Yes**: The subscriber can delete the component.

Developer Deletable
A developer can delete some components after the package is uploaded as Managed - Released. Deleted components aren't removed from Salesforce during a package upgrade. The Protectable attribute contains more details on deleting components.

- **No**: The developer can't delete a Managed - Released component.
- **Yes**: The developer can delete a Managed - Released component.

Protectable
Developers can mark certain components as protected. Protected components can't be linked to or referenced by components created in a subscriber org. A developer can delete a protected component in a future release without worrying about failing installations. However, once a component is marked as unprotected and is released globally, the developer can't delete it. When the subscriber upgrades to a version of the package where the component is deleted, the component is removed from Salesforce.

- **No**: The component can't be marked protected.
- **Yes**: The component can be marked protected.

IP Protection
Certain components automatically include intellectual property protection, such as obfuscating Apex code. The exceptions are Apex methods declared as global, meaning that the subscriber can view the method signatures. Users on AppExchange can view information in the components that you package and publish. Use caution when adding your code to a custom s-control, formula, Visualforce page, or any other component that you can't hide in your app.

- **No**: The component doesn't support intellectual property protection.
- **Yes**: The component supports intellectual property protection.

Component	Upgradeable	Subscriber Deletable	Developer Deletable	Protectable	IP Protection
Action	Yes	No	No	No	No
Analytics Application	No	No	No	No	No
Analytics Dashboard	No	No	No	No	No

Component	Upgradeable	Subscriber Deletable	Developer Deletable	Protectable	IP Protection
Analytics Dataflow	Yes	No	No	No	No
Analytics Dataset	No	No	No	No	No
Analytics Dataset Metadata	No	No	No	No	No
Analytics Lens	No	No	No	No	No
Analytics Recipe	No	No	No	No	No
Workflow Email Alert	Yes	No	Yes, if protected	Yes	No
Apex Class	Yes	No	Yes (if not set to `global` access)	No	Yes
Apex Sharing Reason	Yes	No	No	No	No
Apex Sharing Recalculation	No	Yes	Yes	No	No
Apex Trigger	Yes	No	Yes	No	Yes
Application	Yes	Yes	Yes	No	No
Article Type	Yes	No	No	No	No
Call Center	No	Yes	No	No	No
Compact Layout	Yes	No	No	No	No
Connected App[1]	Yes	Yes	Yes	No	No
Custom Button or Link	Yes	Yes[2]	Yes[3]	No, except custom links (for Home page only)	No
Custom Console Components [1]	Yes	Yes[2]	Yes[3]	No	No
Custom Field on Standard or Custom Object	Yes	Yes[2]	Yes[3]	No	No
Custom Field on Custom Metadata Type	Yes	No	No	No	No
Custom Help Menu Section	Yes	No	No	No	No

[1] Requires a Service Cloud license.

Component	Upgradeable	Subscriber Deletable	Developer Deletable	Protectable	IP Protection
Custom Label	Yes	No	Yes, if protected	Yes	No
Custom Metadata Records	Yes	No	Yes, if protected	Yes	Yes
Custom Metadata Types	Yes	No	No	Yes	Yes
Custom Object	Yes	Yes[2]	Yes[3]	No	No
Custom Permission	Yes	No	No	Yes	No
Custom Report Type	Yes	No	No	No	No
Custom Setting	Yes	Yes[2]	Yes[3]	Yes	Yes
Dashboard	No	Yes	Yes	No	No
Data Classification on Custom Fields	No	Yes	Yes	No	No
Document[7] (10-MB limit)	No	Yes	Yes	No	No
Experience Builder Template	Yes	No	Yes	No	No
Experience Builder Theme	Yes	No	Yes	No	No
External Data Source	Yes	No	No	No	No
Email Template (Classic): Classic	No	Yes	Yes	No	No
Email Template (Lightning): created in Email Template Builder	FlexiPage only	No	No	No	No
Email Template (Lightning): Lightning	No	No	Yes	No	No
Email Template (Lightning): created in Email Template Builder	FlexiPage only	No	No	No	No

Component	Upgradeable	Subscriber Deletable	Developer Deletable	Protectable	IP Protection
External Services[5]	Yes	No	Yes	No	No
Field Set	Yes	Yes[2]	Yes[3]	No	No
Flow	Yes	No	No	No	Yes, except templates
Folder	No	Yes	Yes	No	No
Global Picklist	Yes	Yes	Yes	No	No
Home Page Component	Yes	No	No	No	No
Home Page Layout	No	Yes	Yes	No	No
Inbound Network Connection	Yes	Yes[9]	Yes[9]	No	No
Letterhead	No	Yes	Yes	No	No
Lightning Application	Yes	Yes[8]	Yes[3]	No	No
Lightning Bolt	Yes	No	Yes	No	No
Lightning Component (Aura and Lightning Web Component)	Yes	Yes[8]	Yes[3]	No	No
Lightning Event	Yes	No	No	No	No
Lightning Interface	Yes	No	No	No	No
Lightning page	Yes	No	No	No	No
List View	No	Yes	Yes	No	No
Named Credential[5]	Yes	No	No	No	No
Next Best Action Recommendation Strategy	Yes	No	No	No	Yes, except templates
Outbound Network Connection	Yes	Yes[9]	Yes[9]	No	No
Page Layout	No	Yes	Yes	No	No
Path Assistant	Yes	Yes	Yes	No	No
Permission Set	Yes	Yes[2]	Yes[3]	No	No
Platform Cache[10]	No	No	No	No	No

Component	Upgradeable	Subscriber Deletable	Developer Deletable	Protectable	IP Protection
Platform Event Channel	No	No	No	No	No
Platform Event Channel Member	No	No	No	No	No
Process	See Flow				
Prompts (In-App Guidance)	Yes	No	No	No	No
Record Type	Yes	Yes[2]	Yes[3]	No	No
Remote Site Setting	No	Yes	Yes	No	No
Report	No	Yes	Yes	No	No
Reporting Snapshot	No	Yes	Yes	No	No
Salesforce IoT	Yes	No	No	No	No
S-Control[7] (10-MB limit-)	Yes	Yes[2]	Yes[3]	No	No
Static Resource	Yes	Yes[2]	Yes[3]	No	No
Embedded Service Deployment[6]	No	No	No	No	No
Tab	Yes	Yes[2]	Yes[3]	No	No
TimeSheetTemplate	Yes	Yes[2]	Yes[3]	No	No
Translation	Yes	No	No	No	No
Validation Rule	Yes	Yes[2]	Yes[2]	No	No
Visualforce Component	Yes	Yes[4]	Yes[3]	No	Yes
Visualforce Page	Yes	Yes[2]	Yes[3]	No	No
Workflow Field Update	Yes	No	Yes, if protected	Yes	No
Workflow Outbound Message	Yes	No	Yes, if protected	Yes	No
Workflow Rule	Yes	No	No	No	No
Workflow Task	Yes	No	Yes, if protected	Yes	No

[1] When you remove a connected app that is a component of a package, the app remains available until you update the package. But if you delete the connected app, it's permanently deleted. Any version of the package that contains the deleted connected app is invalidated and can't be installed. You can update a version of the package that *doesn't* contain the connected app as a component. Never delete a connected app that Salesforce distributes, such as the Salesforce app.

[2] If you remove this component type from a new version of your package, the administrator of the subscriber org can delete the component when a subscriber upgrades.

[3] If the ability to remove components is enabled for your packaging org, you can delete these component types, even if they are in a Managed - Released package.

[4] If you remove a public Visualforce component from a new version of your package, the component is removed from the subscriber's org upon upgrade. If the Visualforce component is global, it remains in the subscriber org until the administrator deletes it.

[5] Package developers must add named credential components to the External Services registration package. A subscriber can also create a named credential in Salesforce. However, the subscriber must use the same name as the named credential specified in the External Services registration that references it. Create named credentials manually or with Apex. Be sure to add the named credential to a package so that subscriber orgs can install it. When a subscriber org installs a named credential, it can use the Apex callouts generated by the External Services registration process.

[6] The Salesforce site object isn't packageable. Make sure that the destination org has a site with the same developer name as the site in the source org where the package is created.

[7] The combined size of S-Controls and documents must be less than 10 MB.

[8] When a developer removes an Aura or Lightning web component from a package, the component remains in a subscriber's org after they install the upgraded package. The administrator of the subscriber's org can delete the component, if desired. This behavior is the same for a Lightning web component or an Aura component with a `public` or `global` access value.

[9] You can only delete connections that are in an unprovisioned state.

[10] In API version 51.0 and later, Salesforce provides 3 MB of free Platform Cache capacity for AppExchange-certified and security-reviewed managed packages. This feature is made available through a capacity type called Provider Free capacity and is automatically enabled in Developer Edition orgs.

Component Attributes and Behaviors

Only some attributes of a component are upgradeable. Many components also behave differently or include other restrictions in a managed package. Consider these behaviors when designing your package.

If you register your namespace after you referenced a flow in a Visualforce page or Apex code, don't forget to add the namespace to the flow name. Otherwise, the package will fail to install.

Deleting Visualforce Pages and Global Visualforce Components

Before you delete Visualforce pages or global Visualforce components from your package, remove all references to public Apex classes and public Visualforce components from the pages or components that you're deleting. After removing the references, upgrade your subscribers to an interim package version before you delete the page or global component.

Deleting Lightning Components

We recommend a two-stage process to package developers when you're deleting an Aura component with `global` access or a Lightning web component with an `isExposed` value of `true`. This process ensures that a global component that you delete from the package has no dependencies on the other items in the package.

SEE ALSO:

ISVforce Guide: Deleting Components in Managed Packages

Protected Components

Developers can mark certain components as *protected*. Protected components can't be linked to or referenced by components created in a subscriber org. A developer can delete a protected component in a future release without worrying about failing installations. However, once a component is marked as unprotected and is released globally, the developer can't delete it.

The developer can mark the following components as protected in managed packages.

- Custom labels
- Custom links (for Home page only)
- Custom metadata types
- Custom permissions
- Custom settings
- Workflow alerts
- Workflow field updates
- Workflow outbound messages
- Workflow tasks
- Workflow flow triggers

  The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

Components Automatically Added to Packages

When adding components to your first-generation managed package, related components are automatically added. For example, if you add a Visualforce page to a package that references a custom controller, that Apex class is also added.

To understand what components are automatically included in first-generation managed packages, review the following list:

When you add this component	These components are automatically added
Action	Action target object (if it's a custom object), action target field, action record type, predefined field values, action layout; and any custom fields that the action layout or predefined values refer to on the target object
Reporting Snapshot	Reports
Apex class	Custom fields, custom objects, and other explicitly referenced Apex classes, and anything else that the Apex class references directly Note: If an Apex class references a custom label, and that label has translations, you must explicitly package the individual languages desired for those translations to be included.
Apex trigger	Custom fields, custom objects, and any explicitly referenced Apex classes, and anything else that the Apex trigger references directly
Article type	Custom fields, the default page layout
Compact layout	Custom fields

When you add this component	These components are automatically added
Custom app	Custom tabs (including web tabs), documents (stored as images on the tab), documents folder, asset files
Custom button or link	Custom fields and custom objects
Custom field	Custom objects
Custom home page layouts	Custom home page components on the layout
Custom settings	Apex sharing reasons, Apex sharing recalculations, Apex triggers, custom buttons or links, custom fields, list views, page layouts, record types, validation rules
Custom object	Custom fields, validation rules, page layouts, list views, custom buttons, custom links, record types, Apex sharing reasons, Apex sharing recalculations, and Apex triggers Note: • Apex sharing reasons are unavailable in extensions. • When packaged and installed, only public list views from an app are installed. If a custom object has any custom list views that you want to include in your package, ensure that the list view is accessible by all users.
Custom object (as an external object)	External data source, custom fields, page layouts, list views, custom buttons, and custom links Note: • When packaged and installed, only public list views from an app are installed. If an external object has any custom list views that you want to include in your package, ensure that the list view is accessible by all users. • In managed and unmanaged packages, external objects are included in the custom object component.
Custom tab	Custom objects (including all of its components), s-controls, and Visualforce pages
Dashboard	Folders, reports (including all of its components), s-controls, and Visualforce pages
Document	Folder
Email template (Classic)	• Folder • Letterhead • Custom fields • Documents (stored as images on the letterhead or template)
Email template (Lightning)	• Custom object • Custom field references (in Handlebars Merge Language syntax) • Enhanced folder (except public and private folders) • Inline images referencing Salesforce Files • Attachments referencing Salesforce Files

When you add this component	These components are automatically added
	For pre-existing Lightning email templates, attachments aren't automatically added to the package. Open and resave these templates to turn the attachments into content assets, which are then automatically added to the package These items can't be added to a Lightning email template package: • Enhanced letterhead • The associated FlexiPage • CMS files (Pardot only)
Email template (Lightning) created in Email Template Builder	• Custom object • Custom field references (in Handlebars Merge Language syntax) • Enhanced folder (except public and private folders) • Attachments referencing Salesforce Files • The associated FlexiPage For pre-existing Email Template Builder email templates, attachments aren't automatically added to the package. Open and resave these templates to turn the attachments into content assets, which are then automatically added to the package These items can't be added to an Email Template Builder email template package: • Enhanced letterhead • Inline images referencing Salesforce Files • CMS files (Pardot only)
Field set	Any referenced fields
Lightning page	All Lightning resources referenced by the page, such as record types, actions, custom components, events, and interfaces. Custom fields, custom objects, list views, page layouts, Visualforce pages, and Apex classes referenced by the components on the page.
Lightning page tab	Lightning page
Flow	Custom objects, custom fields, Apex classes, and Visualforce pages
Folder	Everything in the folder
Lightning application	All Lightning resources referenced by the application, such as components, events, and interfaces. Custom fields, custom objects, list views, page layouts, and Apex classes referenced by the application.
Lightning component	All Lightning resources referenced by the component, such as nested components, events, and interfaces. Custom fields, custom objects, list views, page layouts, and Apex classes referenced by the component.
Lightning event	Custom fields, custom objects, list views, and page layouts
Lightning interface	Custom fields, custom objects, list views, and page layouts

When you add this component	These components are automatically added
Lightning web component	All Lightning web component resources referenced by the component, such as nested components and modules. Custom fields, custom objects, list views, page layouts, and Apex classes referenced by the component.
Page layout	Actions, custom buttons, custom links, s-controls, and Visualforce pages
Permission set	Any custom permissions, external data sources, Visualforce pages, record types, and Apex classes that are assigned in the permission set
Record type	Record type mappings, compact layout
Report	Folder, custom fields, custom objects, custom report types, and custom s-controls
S-control	Custom fields and custom objects
Translation	Translated terms for the selected language on any component in the package
Validation rule	Custom fields (referenced in the formula)
Visualforce home page component	Associated Visualforce page
Visualforce pages	Apex classes that are used as custom controllers, Visualforce custom components, and referenced field sets
Workflow rule	All associated workflow alerts, field updates, outbound messages, and tasks; also, if the workflow rule is designed for a custom object, the custom object is automatically included

Note: Some package components, such as validation rules or record types, don't appear in the list of package components, but are included and install with the other components.

Editing Components and Attributes After Installation

The following table shows which components and attributes are editable after installation from a managed package.

Developer Editable
The developer can edit the component attributes in this column. These attributes are locked in the subscriber's organization.

Subscriber and Developer Editable
The subscriber and developer can edit the component attributes in this column. However, these attributes aren't upgradeable. Only new subscribers receive the latest changes.

Locked
After a package is Managed - Released, the developer and subscriber can't edit the component attributes in this column.

Component	Developer Editable	Subscriber and Developer Editable	Locked
Action		• Action layout • Predefined values for action fields	• All fields
Apex Class	• API Version		• Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
	• Code		
Apex Sharing Reason	• Reason Label		• Reason Name
Apex Sharing Recalculation		• Apex Class	
Apex Trigger	• API Version • Code		• Name
Application	• Show in Lightning Experience (SalesforceClassic only) • Selected Items (Lightning Experience only) • Utility Bar (Lightning Experience only)	• All attributes, except App Name and Show in Lightning Experience (Salesforce Classic only) • All attributes, except Developer Name, Selected Items, and Utility Bar (Lightning Experience only)	• App Name (SalesforceClassic only) • Developer Name (Lightning Experience only)
Article Types	• Description • Label • Plural Label • Starts with a Vowel Sound	• Available for Customer Portal • Channel Displays • Default Sharing Model • Development Status • Enable Divisions • Grant Access Using Hierarchy • Search Layouts	• Name
Compact Layout		• All attributes	
Connected App	• Access Method • Canvas App URL • Callback URL • Connected App Name • Contact Email • Contact Phone • Description • Icon URL • Info URL • Trusted IP Range	• ACS URL • Entity ID • IP Relaxation • Manage Permission Sets • Manage Profiles • Mobile Start URL • Permitted Users • Refresh Token Policy • SAML Attributes • Service Provider Certificate	• API Name • Created Date/By • Consumer Key • Consumer Secret • Installed By • Installed Date • Last Modified Date/By • Version

Component	Developer Editable	Subscriber and Developer Editable	Locked
	• Locations • Logo Image URL • OAuth Scopes	• Start URL • Subject Type	
Custom Button or Link	• Behavior • Button or Link URL • Content Source • Description • Display Checkboxes • Label • Link Encoding	• Height • Resizeable • Show Address Bar • Show Menu Bar • Show Scrollbars • Show Status Bar • Show Toolbars • Width • Window Position	• Display Type • Name
Custom Field	• Auto-Number Display Format • Decimal Places • Description • Default Value • Field Label • Formula • Length • Lookup Filter • Related List Label • Required • Roll-Up Summary Filter Criteria	• Chatter Feed Tracking • Help Text • Mask Type • Mask Character • Sharing Setting • Sort Picklist Values • Track Field History	• Child Relationship Name • Data Type • External ID • Field Name • Roll-Up Summary Field • Roll-Up Summary Object • Roll-Up Summary Type • Unique
Custom Label	• Category • Short Description • Value		• Name
Custom Object	• Description • Label • Plural Label • Record Name • Starts with a Vowel Sound	• Allow Activities • Allow Reports • Available for Customer Portal • Context-Sensitive Help Setting • Default Sharing Model	• Object Name • Record Name Data Type • Record Name Display Format

Component	Developer Editable	Subscriber and Developer Editable	Locked
		• Development Status • Enable Divisions • Enhanced Lookup • Grant Access Using Hierarchy • Search Layouts • Track Field History	
Custom Permission	• Connected App • Description • Label • Name		
Custom Report Type	• All attributes except Development Status and Report Type Name	• Development Status	• Report Type Name
Custom Setting	• Description • Label		• Object Name • Setting Type • Visibility
Dashboard		• All attributes except Dashboard Unique Name	• Dashboard Unique Name
Document		• All attributes except Document Unique Name	• Document Unique Name
Email Template: Classic		• All attributes except Email Template Name	• Email Template Name
Email Template: Lightning			All attributes
Email Template: Email Template Builder			All attributes
External Data Source	• Type	• Auth Provider • Certificate • Custom Configuration • Endpoint • Identity Type • OAuth Scope • Password	• Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
		• Protocol • Username	
Field Set	• Description • Label • Available fields	• Selected fields (only subscriber controlled)	• Name
Flow	• Entire flow	• Flow Label • Description • Status	• Flow API Name • URL
Folder		• All attributes except Folder Unique Name	• Folder Unique Name
Home Page Component	• Body • Component Position		• Name • Type
Home Page Layout		• All attributes except Layout Name	• Layout Name
Inbound Network Connection	• Connection Type • Developer Name • Description • Master Label • Region	• Status	
Letterhead		• All attributes except Letterhead Name	• Letterhead Name
Lightning Application	• API Version • Description • Label • Markup		Name
Lightning Component	• API Version • Description • Label • Markup		Name
Lightning Event	• API Version		Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
	• Description • Label • Markup		
Lightning Interface	• API Version • Description • Label • Markup		Name
Lightning Page	• Lightning page		
Lightning Web Component	• API Version • Description • Label • Markup		Name
List View		• All attributes except View Unique Name	• View Unique Name
Named Credential	• Endpoint • Label	• All attributes except Endpoint, Label, and Name	• Name
Outbound Network Connection	• Connection Type • Developer Name • Description • Master Label • Region • Service Name	• Status	
Page Layout		• All attributes except Page Layout Name	• Page Layout Name
Path Assistant		IsActive field	SobjectType, SobjectProcessField, and RecordType
Permission Set	• Description • Label • Custom object permissions • Custom field permissions		• Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
	• Apex class access settings • Visualforce page access settings		
Platform Cache	• Master Label • Description • Default Partition	• Organization Capacity • Trial Capacity	• Developer Name
Record Type	• Description • Record Type Label	• Active • Business Process	• Name
Remote Site Setting		All attributes except Remote Site Name	• Remote Site Name
Report		• All attributes except Report Unique Name	• Report Unique Name
Reporting Snapshot		• All attributes except Reporting Snapshot Unique Name	• Reporting Snapshot Unique Name
S-Control	• Content • Description • Encoding • Filename • Label	• Prebuild in Page	• S-Control Name • Type
Static Resource	• Description • File		• Name
Tab	• Description • Encoding • Has Sidebar • Height • Label • S-control • Splash Page Custom Link • Type • URL • Width	• Tab Style	• Tab Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
Translation	• All attributes		
Validation Rule	• Description • Error Condition Formula • Error Location • Error Message	• Active	• Rule Name
Visualforce Component	• API Version • Description • Label • Markup		• Name
Visualforce Page	• API Version • Description • Label • Markup		• Name
Workflow Email Alert		• Additional Emails • Email Template • From Email Address • Recipients	• Description
Workflow Field Update	• Description • Field Value • Formula Value	• Lookup	• Name
Workflow Outbound Message	• Description • Endpoint URL • Fields to Send • Send Session ID	• User to Send As	• Name
Workflow Rule	• Description • Evaluation Criteria • Rule Criteria	• Active	• Rule Name
Workflow Task		• Assign To • Comments • Due Date • Priority	• Subject

Component	Developer Editable	Subscriber and Developer Editable	Locked
		• Record Type • Status	

Component Behavior in Packages

Determine which components to include in your package and understand how components impact app design and distribution requirements for your managed or unmanaged packages.

EDITIONS

Available in: Salesforce Classic

Available in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create AppExchange packages:
- Create AppExchange Packages

Note:

- For more information on the attributes and properties of each component in packages, see the ISVforce Guide.
- Component names must be unique within an org. To ensure that your component names don't conflict with component names in an installer's org, use a managed package. The managed package ensures that all your component names contain your namespace prefix.

Apex Classes or Triggers

Any Apex that is included as part of a package must have at least 75% cumulative test coverage. Each trigger must also have some test coverage. When you upload your package to AppExchange, all tests are run to ensure that they run without errors. In addition, all tests are run when the package is installed in the installer's org. If any test fails, the installer can decide whether to install the package.

Tip: To prevent naming conflicts, Salesforce recommends using managed packages for all packages that contain Apex to ensure that all Apex objects contain your namespace prefix. For example, if an Apex class is called `MyHelloWorld` and your org's namespace is `OneTruCode`, the class is referenced as `OneTruCode.MyHelloWorld`.

Keep the following considerations in mind when including Apex in your package.

- Managed packages receive a unique namespace. This namespace is prepended to your class names, methods, variables, and so on, which helps prevent duplicate names in the installer's org.
- In a single transaction, you can only reference 10 unique namespaces. For example, suppose that you have an object that executes a class in a managed package when the object is updated. Then that class updates a second object, which in turn executes a different class in a different package. Even though the first package didn't access the second package directly, the access occurs in the same transaction. It's therefore included in the number of namespaces accessed in a single transaction.
- If you're exposing any methods as Web services, include detailed documentation so that subscribers can write external code that calls your Web service.
- If an Apex class references a custom label and that label has translations, explicitly package the individual languages desired to include those translations in the package.
- If you reference a custom object's sharing object (such as MyCustomObject__share) in Apex, you add a sharing model dependency to your package. Set the default org-wide access level for the custom object to Private so other orgs can install your package successfully.
- The code contained in an Apex class, trigger, or Visualforce component that's part of a managed package is obfuscated and can't be viewed in an installing org. The only exceptions are methods declared as global. You can view global method signatures

in an installing org. In addition, License Management Org users with the View and Debug Managed Apex permission can view their packages' obfuscated Apex classes when logged in to subscriber orgs via the Subscriber Support Console.

- You can use the `deprecated` annotation in Apex to identify `global` methods, classes, exceptions, enums, interfaces, and variables that can't be referenced in later releases of a managed package. So you can refactor code in managed packages as the requirements evolve. After you upload another package version as Managed - Released, new subscribers that install the latest package version can't see the deprecated elements, while the elements continue to function for existing subscribers and API integrations.
- Any Apex contained in an unmanaged package that explicitly references a namespace can't be uploaded.
- Apex code that refers to Data Categories can't be uploaded.
- Before you delete Visualforce pages or global Visualforce components from your package, remove all references to public Apex classes and public Visualforce components from the pages or components that you're deleting. After removing the references, upgrade your subscribers to an interim package version before you delete the page or global component.

Apex Sharing Reasons

Apex sharing reasons can be added directly to a package, but are only available for custom objects.

Connected Apps

- Connected apps can be added to managed packages, only. Connected apps are not supported for unmanaged packages.
- Subscribers or installers of a package can't delete a connected app by itself; they can only uninstall its package. A developer can delete a connected app after a package is uploaded as Managed - Released. The connected app is deleted in the subscriber's org during a package upgrade.
- If you update a connected app and include it in a new package version, upgrading that package in a customer org updates the existing connected app.
- If you push upgrade a package containing a connected app whose OAuth scope or IP ranges have changed from the previous version, the upgrade fails. This security feature blocks unauthorized users from gaining broad access to a customer org by upgrading an installed package. A customer can still perform a pull upgrade of the same package. This upgrade is allowed because it's with the customer's knowledge and consent.
- You can add an existing connected app (one created before Summer '13) to a managed package. You can also combine new and existing connected apps in the same managed package.
- For connected apps created before Summer '13, the existing install URL is valid until you package and upload a new version. After you upload a new version of the package with an updated connected app, the install URL no longer works.

Custom Console

A package that has a custom console component can only be installed in an org with the Service Cloud license or Sales Console permission enabled.

Custom Fields

- Developers can add required and universally required custom fields to managed packages as long as they have default values.
- Auto-number type fields and required fields can't be added after the object is uploaded in a Managed - Released package.
- Subscriber orgs can't install roll-up summary fields that summarize detail fields set to *protected*.

Custom Labels

If a label is translated, the language must be explicitly included in the package for the translations to be included in the package. Subscribers can override the default translation for a custom label.

Custom Metadata Types

Second-generation managed packages (2GP) include the fields and records for custom metadata types that you add. You can't add fields directly to an existing package after the package version is promoted. If you create multiple packages that share a namespace, then layouts and records can be in separate packages, but custom fields on the custom metadata type must be in the same package.

You can add fields to a custom metadata type by publishing an extension to the existing package, creating an entity relationship field, and mapping the field to the custom metadata type in your extension. See Add Custom Metadata Type Fields to Existing Packages.

Custom Objects

- If a developer enables the `Allow Reports` or `Allow Activities` attributes on a packaged custom object, the subscriber's org also has these features enabled during an upgrade. After it's enabled in a Managed - Released package, the developer and the subscriber can't disable these attributes.
- Standard button and link overrides are also packageable.
- In your extension package, if you want to access history information for custom objects contained in the base package, work with the base package owner to:

  1. Enable history tracking in the release org of the base package.
  2. Upload a new version of the base package.
  3. Install the new version of the base package in the release org of the extension package to access the history tracking info.

  As a best practice, don't enable history tracking for custom objects contained in the base package directly in the extension package's release org. Doing so can result in an error when you install the package and when you create patch orgs for the extension package.

Custom Permissions

If you deploy a change set with a custom permission that includes a connected app, the connected app must already be installed in the destination org.

Custom Report Types

A developer can edit a custom report type in a managed package after it's released, and can add new fields. Subscribers automatically receive these changes when they install a new version of the managed package. However, developers can't remove objects from the report type after the package is released. If you delete a field in a custom report type that's part of a managed package, and the deleted field is part of bucketing or used in grouping, an error message appears.

Custom Settings

- If a custom setting is contained in a managed package, and the `Visibility` is specified as Protected, the custom setting isn't contained in the list of components for the package on the subscriber's org. All data for the custom setting is hidden from the subscriber.

Custom Tabs

- The tab style for a custom tab must be unique within your app. However, it doesn't have to be unique within the org where it's installed. A custom tab style doesn't conflict with an existing custom tab in the installer's environment.
- To provide custom tab names in different languages, from Setup, enter *`Rename Tabs and Labels`* in the Quick Find box, then select **Rename Tabs and Labels**.
- Subscribers can't edit custom tabs in a managed package.

Customer Portal and Partner Portal

Packages referring to Customer Portal or partner portal fields are supported. The subscriber installing the package must have the respective portal enabled to install the package.

Dashboard Components

Developers of managed packages must consider the implications of introducing dashboard components that reference reports released in a previous version of the package. If the subscriber deleted the report or moved the report to a personal folder, the dashboard component referencing the report is dropped during the installation. Also, if the subscriber has modified the report, the report results can impact what displays in the dashboard component. As a best practice, release a dashboard and the related reports in the same version.

Divisions

- When divisions are enabled on a custom object in a package, the subscribing org must have the divisions feature enabled to install the package.
- Setting the division filter on a report doesn't cause a dependency. The setting is dropped when installed into the subscriber's org.
- Summarizing by the object's division field—for example, Account Division—in a report causes a dependency.
- If the object's division field in a report is included as a column, and the subscriber's org doesn't support divisions on the object, the column is dropped during installation.
- If you install a custom report type that includes an object's division field as a column, that column is dropped if the org doesn't support divisions.

Lightning Email Templates

Create managed packages using the first-generation packaging tool.

To add an email template created in Email Template Builder to a managed package, you need the **Access Drag-and-Drop Content Builder** user permission.

These packaging considerations apply to Lightning email templates, including email templates created in Email Template Builder.

- An email template created in Email Template Builder that contains inline images with references to Salesforce Files can't be added to packages or change sets.
- For email templates created in Email Template Builder before the Spring '21 release, attachments aren't automatically added to the package. Open and resave these templates to turn the attachments into content assets, which are then automatically added to the package.
- Enhanced email template folders have these behaviors:
  - If a package includes an enhanced email template folder, the target organization must have enhanced folders enabled for the deploy to succeed.
  - If an email template is in a subfolder, adding the root folder to a package doesn't automatically add the email template to the package. If the email template is in the root folder, it's automatically added to the package.
- For merge fields based on custom fields that are used in the Recipients prefix (for leads and contacts), we add references to those merge fields. If the custom field is renamed, the reference in the template isn't updated. Edit the custom merge field to use the new field name and update the reference.

External Data Sources

- After installing an external data source from a managed or unmanaged package, the subscriber must reauthenticate to the external system.
  - For password authentication, the subscriber must reenter the password in the external data source definition.
  - For OAuth, the subscriber must update the callback URL in the client configuration for the authentication provider, then reauthenticate by selecting `Start Authentication Flow on Save` on the external data source.
- Certificates aren't packageable. If you package an external data source that specifies a certificate, make sure that the subscriber org has a valid certificate with the same name.

External Objects

- In managed and unmanaged packages, external objects are included in the custom object component.
- Include External Change Data Tracking components in a managed package by selecting your test from the Apex Class Component Type list. The trigger, test, external data source, external object, and other related assets are brought into the package for distribution.

Field Dependencies

- Developers and subscribers can add, change, or remove field dependencies.
- If the developer adds a field dependency, it's added during installation unless the subscriber has already specified a dependency for the same field.
- If a developer removes a dependency, this change isn't reflected in the subscriber's org during an upgrade.
- If the developer introduces a new picklist value mapping between the dependent and controlling fields, the mapping is added during an upgrade.
- If a developer removes a picklist value mapping, the change isn't reflected in the subscriber's org during an upgrade.

Field Sets

Field sets in installed packages perform different merge behaviors during a package upgrade:

If a package developer:	Then in the package upgrade:
Changes a field from **Unavailable** to **Available for the Field Set** or **In the Field Set**	The modified field is placed at the end of the upgraded field set in whichever column it was added to.
Adds a field	The new field is placed at the end of the upgraded field set in whichever column it was added to.
Changes a field from **Available for the Field Set** or **In the Field Set** to **Unavailable**	The field is removed from the upgraded field set.
Changes a field from **In the Field Set** to **Available for the Field Set** (or vice versa)	The change isn't reflected in the upgraded field set.

Note: Subscribers aren't notified of changes to their installed field sets. The developer must notify users ——of changes to released field sets through the package release es or other documentation. Merging has the potential to remove fields in your field set.

When a field set is installed, a subscriber can add or remove any field.

Flows

- When you upload a package or package version, the active flow version is included. If the flow has no active version, the latest version is packaged.
- To update a managed package with a different flow version, activate that version and upload the package again. Or deactivate all versions of the flow, make sure the latest flow version is the one to distribute, and then upload the package.
- In a development org, you can't delete a flow or flow version after you upload it to a released or beta managed package.
- You can't delete flows from Managed - Beta package installations in development org.
- You can't delete a flow from an installed package. To remove a packaged flow from your org, deactivate it and then uninstall the package.
- If you have multiple versions of a flow installed from multiple unmanaged packages, you can't remove only one version by uninstalling its package. Uninstalling a package—managed or unmanaged—that contains a single version of the flow removes the entire flow, including all versions.
- You can't include flows in package patches.
- An active flow in a package is active after it's installed. The previous active version of the flow in the destination org is deactivated in favor of the newly installed version. Any in-progress flows based on the now-deactivated version continue to run without interruption but reflect the previous version of the flow.

- Upgrading a managed package in your org installs a new flow version only if there's a newer flow version from the developer. After several upgrades, you can end up with multiple flow versions.
- If you install a managed package that contains multiple flow versions in a fresh destination org, only the latest flow version is deployed.
- If you install a flow from an unmanaged package that has the same name but a different version number as a flow in your org, the newly installed flow becomes the latest version of the existing flow. However, if the packaged flow has the same name and version number as a flow already in your org, the package install fails. You can't overwrite a flow.
- Flow Builder can't open flows that are installed from managed packages, unless they're templates.
- You can't create a package that contains flows invoked by both managed and unmanaged package pages. As a workaround, create two packages, one for each type of component. For example, suppose that you want to package a customizable flow invoked by a managed package page. Create one unmanaged package with the flow that users can customize. Then create another managed package with the Visualforce page referencing the flow (including namespace) from the first package.
- When you translate a flow from a managed package, the flow's Master Definition Name doesn't appear on the Translate page or the Override page. To update the translation for the Master Definition Name, edit the flow label and then update the translation from the Translate page.
- If any of the following elements are used in a flow, packageable components that they reference aren't included in the package automatically. To deploy the package successfully, manually add those referenced components to the package.
  - Post to Chatter
  - Send Email
  - Submit for Approval
- If a flow references a Lightning component that depends on a CSP Trusted Site, the trusted site isn't included in the package or change set automatically.

Folders

- Components that Salesforce stores in folders, such as documents, can't be added to packages when stored in personal and unfiled folders. Put documents, reports, and other components that Salesforce stores in folders in one of your publicly accessible folders.
- Components such as documents, email templates, reports, or dashboards are stored in new folders in the installer's org using the publisher's folder names. Give these folders names that indicate they're part of the package.
- If a new report, dashboard, document, or email template is installed during an upgrade, and the folder containing the component was deleted by the subscriber, the folder is re-created. Any components in the folder that were previously deleted aren't restored.
- The name of a component contained in a folder must be unique across all folders of the same component type, excluding personal folders. Components contained in a personal folder must be unique within the personal folder only.

Home Page Components

When you package a custom home page layout, all the custom home page components included on the page layout are automatically added. Standard components such as Messages & Alerts aren't included in the package and don't overwrite the installer's Messages & Alerts. To include a message in your custom home page layout, create an HTML Area type custom Home tab component containing your message. From Setup, enter *Home Page Components* in the Quick Find box, then select **Home Page Components**. Then add the message to your custom home page layout.

Home Page Layouts

After they're installed, your custom home page layouts are listed with all the subscriber's home page layouts. Distinguish them by including the name of your app in the page layout name.

Inbound Network Connections

- Packaged connections are installed as unprovisioned. Alert subscribers about how to provision connections after package installation.

- If a developer changes the Region of a packaged connection that is subscriber-provisioned, the upgrade fails for the subscriber. Alert subscribers about tearing down the connection before updating the Region field. As a best practice, avoid changing the Region of a packaged connection unless necessary.

List Views

List views associated with queues can't be included in a package.

Multi-Currency

- If a subscriber installs a report or custom report type that includes an object's currency field as a column, that column is dropped if the subscriber's org isn't enabled for multiple currencies.
- Referencing an object's currency field in a report's criteria—for example, `Account Currency`—causes a dependency.
- Summarizing by an object's currency field in a report causes a dependency.
- Using a currency designation in a report criteria value—for example, "Annual Revenue equals GBP 100"—doesn't cause a dependency. The report generates an error when run in the installers org if it doesn't support the currency.
- If an object's currency field in a report is included as a column and the subscriber's org isn't enabled for multiple currencies, that column is dropped during installation.
- If a subscriber installs a custom report type that includes an object's currency field as a column, that column is dropped if the org isn't enabled for multiple currencies.

Named Credentials

- After installing a named credential from a managed or unmanaged package, the subscriber must reauthenticate to the external system.
  - For password authentication, the subscriber reenters the password in the named credential definition.
  - For OAuth, the subscriber updates the callback URL in the client configuration for the authentication provider and then reauthenticates by selecting **Start Authentication Flow on Save** on the named credential.
- Named credentials aren't automatically added to packages. If you package an external data source or Apex code that specifies a named credential as a callout endpoint, add the named credential to the package. Alternatively, make sure that the subscriber org has a valid named credential with the same name.

  If you have multiple orgs, you can create a named credential with the same name but with a different endpoint URL in each org. You can then package and deploy—on all the orgs—one callout definition that references the shared name of those named credentials. For example, the named credential in each org can have a different endpoint URL to accommodate differences in development and production environments. If an Apex callout specifies the shared name of those named credentials, the Apex class that defines the callout can be packaged and deployed on all those orgs without programmatically checking the environment.
- Certificates aren't packageable. If you package a named credential that specifies a certificate, make sure that the subscriber org has a valid certificate with the same name.

Outbound Network Connections

- Packaged connections are installed as unprovisioned. Alert subscribers about how to provision connections after package installation.
- If a developer changes the Region or Service Name of a packaged connection that is subscriber-provisioned, the upgrade fails for the subscriber. Alert subscribers about tearing down the connection before you update the Region or Service Name fields. As a best practice, avoid changing the Region or Service Name of a packaged connection unless necessary.
- If you package a Named Credential that references an Outbound Network Connection, the referenced Outbound Network Connection component is automatically added to the package.

Page Layouts

The page layout of the person uploading a package is the layout used for Group and Professional Edition orgs and becomes the default page layout for Enterprise, Unlimited, Performance, and Developer Edition orgs.

Package page layouts alongside complimentary record types if the layout is being installed on an existing object. Otherwise, manually apply the installed page layouts to profiles.

If a page layout and a record type are created as a result of installing a package, the uploading user's page layout assignment for that record type is assigned to that record type for all profiles in the subscriber org, unless a profile is mapped during an install or upgrade.

Permission Sets

You can include permission sets as components in a package, with the following permissions and access settings:

- Assigned custom apps
- Custom object permissions
- External object permissions
- Custom field permissions
- Custom permissions
- Custom tab visibility settings
- Apex class access
- Visualforce page access
- External data source access

Note: Standard tab visibility settings aren't included in permission set components.

Use permission sets to install or upgrade a collection of permissions. In contrast to profile settings, permission sets don't overwrite profiles.

Picklist Values

- When explicitly referencing a picklist value in code, keep in mind that picklist values for custom fields can be renamed, added, edited, or deleted by subscribers. Carefully consider this possibility when explicitly referencing a picklist value in code.
- Picklist field values can be added or deleted in the developer's organization.
- Changes to standard picklists can't be packaged and deployed to subscriber orgs and picklist values deleted by the developer are still available in the subscriber's org. If there are differences between the package and the target org, or if there are dependencies on new values from features such as PathAssistant, the deploy fails. To change values in subscriber orgs, you must manually add or modify the values in the target subscriber org.
- Updating picklist values in unlocked packages isn't supported. Manually add or modify the values in the target subscriber org.
- Package upgrades retain dependent picklist values that are saved in a managed custom field.
- Global value sets can be added to developer and subscriber orgs. Global value sets have the following behavior during a package upgrade:
  - Label and API names for field values don't change in subscriber orgs.
  - New field values aren't added to the subscriber orgs.
  - Active and inactive value settings in subscriber orgs don't change.
  - Default values in subscriber orgs don't change.
  - Global value set label names change if the package upgrade includes a global value set label change.

Profile Settings

Profile settings include the following for components in the package:

- Assigned custom apps
- Assigned connected apps

- Tab settings
- Page layout assignments
- Record type assignments
- Custom field permissions
- Custom metadata type permissions
- Custom object permissions
- Custom permissions
- Custom settings permissions
- External object permissions
- Apex class access
- Visualforce page access
- External data source access

Profile settings overwrite existing profiles in the installer's org with specific permission and setting changes. Profile Settings get applied only if the package is installed in the target org for specific profiles and not if it's installed for all profiles.

Record Types

- If record types are included in the package, the subscriber's org must support record types to install the package.
- When a new picklist value is installed, it's associated with all installed record types according to the mappings specified by the developer. A subscriber can change this association.
- Referencing an object's record type field in a report's criteria—for example, `Account Record Type`—causes a dependency.
- Summarizing by an object's record type field in a report's criteria—for example, `Account Record Type`—causes a dependency.
- If an object's record type field is included as a column in a report, and the subscriber's org isn't using record types on the object or doesn't support record types, the column is dropped during installation.
- If you install a custom report type that includes an object's record type field as a column, that column is dropped if the org doesn't support record types or the object doesn't have record types defined.

Reporting Snapshots

Developers of managed packages must consider the implications of introducing reporting snapshots that reference reports released in a previous version of the package. If the subscriber deleted the report or moved the report to a personal folder, the reporting snapshot referencing the report isn't installed, even though the Package Installation page indicates that it will be. Also, if the subscriber has modified the report, the report can return results impacting the information displayed by the reporting snapshot. As a best practice, the developer releases the reporting snapshot and the related reports in the same version.

Because the subscriber selects the running use, some reporting snapshot field mappings could become invalid if the running user doesn't have access to source or target fields.

Reports

If a report includes elements that can't be packaged, those elements are dropped or downgraded, or the package upload fails. For example:

- Hierarchy drill-downs are dropped from activity and opportunities reports.
- Filters on unpackageable fields are automatically dropped (for example, in filters on standard object record types).
- Package upload fails if a report includes filter logic on an unpackageable field (for example, in filters on standard object record types).
- Lookup values on the `Select Campaign` field of standard campaign reports are dropped.
- Reports are dropped from packages if they've been moved to a private folder or to the Unfiled Public Reports folder.

- When a package is installed into an org that doesn't have Chart Analytics 2.0:
  - Combination charts are downgraded instead of dropped. For example, a combination vertical column chart with a line added is downgraded to a simple vertical column chart; a combination bar chart with more bars is downgraded to a simple bar chart.
  - Unsupported chart types, such as donut and funnel, are dropped.

S-Controls

Only s-controls in unmanaged packages created before January 2010 can be installed by subscribers.

S-controls have been deprecated and are superseded by Visualforce pages.

Translation Workbench

- If you've enabled the translation workbench and added a language to your package, any associated translated values are automatically packaged for the appropriate components in your package. Make sure that you have provided translations for all possible components.
- An installer of your package can see which languages are supported on the package detail page. The installer doesn't need to enable anything to have the packaged language translations appear. The only reasons installers might want to enable the translation workbench are to change translations for unmanaged components after installation, override custom label translations in a managed package, or translate into more languages.
- If you're designing a package extension, you can include translations for the extension components but not translations for components in the base package.

Validation Rules

For custom objects that are packaged, any associated validation rules are implicitly packaged as well.

Analytics

Analytics components include Analytics applications, dashboards, dataflows, datasets, lenses, recipes, and user XMD. As you package Analytics components, keep these tips and best practices in mind.

- Analytics unmanaged packages, as opposed to managed packages, are considered a developer-only feature and aren't supported for general-purpose distribution. While Analytics unmanaged packages work as expected within the constraints of Salesforce unmanaged packages, they aren't subject to as much testing as managed packages. Unmanaged packages come without many of the safeguards of managed packages, and are intended for developers familiar with their limitations. Also refer to the relevant topic in the ISV Guide.
- Before a recipe is available for packaging, you must create a dataset with the recipe. The related dataflow must be added to the package along with the recipe for deployment to succeed.
- Analytics Admin permissions are required to create a package but not for deployment, which requires only Salesforce admin permissions.
- There's no spidering between datasets and dataflows, meaning there's no dependency following. When packaging both, they must be added manually. If they aren't, an error appears during deployment. The same is true for change sets—when packaging both datasets and dataflows, add them manually.
- When you package a data flow, source and security predicates aren't included in the package.
- Because views are user-specific, they aren't included when you package the dashboard.
- If you migrate dashboards manually using JSON copy/paste, any conditional formatting, widget-specific number formats, and measure labels on blended queries are lost. To retain these formats and labels in the migrated dashboard, include the Analytics Dataset Metadata component type when packaging your change set.
- The Winter '18 release contains a beta version of Apex steps, which lets developers include custom Apex functionality in a dashboard to access Salesforce platform features that aren't inherently supported in Analytics. If you include dashboards in a package, Apex steps aren't included—migrate Apex classes separately.

- Before the Spring '17 release, images didn't render when deploying a dashboard that used an image widget that referenced image files not available on the target org. There were two workarounds: Manually upload the images, or add a folder containing the images to the package. As of the Spring '17 release, images are packaged with the dashboard, and references between dashboards are maintained. You can't delete a dashboard that is referenced in a link. Either re-create the image, or link the widgets in the dashboard in the source org. Then repackage or fix the link issues in the target org.
- Take care when packaging dataflows. Invalid schema overrides and unsupported or illegal parameters are removed. For example, `Type = dim` is no longer supported. Use `Type = text` instead. Comments in JSON are removed. Nodes can appear in a different order.

Workflow

- Salesforce prevents you from uploading workflow alerts that have a public group, partner user, or role recipient. Change the recipient to a user before uploading your app. During installation, Salesforce replaces that user with the user installing the app, and the installer can customize it as necessary.
- Salesforce prevents you from uploading workflow field updates that change an `Owner` field to a queue. Change the updated field value to a user before uploading your app. During installation, Salesforce replaces that user with the user installing the app, and the installer can customize it as necessary.
- Salesforce prevents you from uploading workflow rules, field updates, and outbound messages that reference a record type on a standard or managed-installed object.
- Salesforce prevents you from uploading workflow tasks that are assigned to a role. Change the `Assigned To` field to a user before uploading your app. During installation, Salesforce replaces that user with the user installing the app, and the installer can customize it as necessary.
- You can package workflow rules and associated workflow actions, such as email alerts and field updates. However, any time-based triggers aren't included in the package. Notify your installers to set up any time-based triggers that are essential to your app.

  Flow triggers aren't packageable. The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.
- Developers can protect some workflow actions.
- Developers can associate or disassociate workflow actions with a workflow rule at any time. These changes, including disassociation, are reflected in the subscriber's org upon install. In managed packages, a subscriber can't disassociate workflow actions from a workflow rule if it was associated by the developer.
- References to a specific user in workflow actions, such as the email recipient of a workflow email alert, are replaced by the user installing the package. Workflow actions referencing roles, public groups, account team, opportunity team, or case team roles may not be uploaded.
- References to an org-wide address, such as the `From email address` of a workflow email alert, are reset to Current User during installation.
- On install, all workflow rules newly created in the installed or upgraded package, have the same activation status as in the uploaded package.

Protected Components

Developers can mark certain components as *protected*. Protected components can't be linked to or referenced by components created in a subscriber org. A developer can delete a protected component in a future release without worrying about failing installations. However, after a component is marked as unprotected and is released globally, the developer can't delete it. Developers can mark the following components as protected in managed packages:

- Custom labels
- Custom links (for Home page only)

- Workflow alerts
- Workflow field updates
- Workflow flow triggers

  The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

- Workflow outbound messages
- Workflow tasks

Intellectual Property Considerations

The following information is important when considering your intellectual property and its protection.

- Only publish package components that are your intellectual property and that you have the rights to share.
- After components are available on AppExchange, you can't recall them from anyone who has installed them.
- The information in the components you package and publish may be visible to users on AppExchange. Use caution when adding your code to a formula Visualforce page, or any other component that you can't hide in your app.
- The code contained in Apex that is part of a managed package is automatically obfuscated and can't be viewed in an installing org. The only exceptions are methods declared as global, meaning that the method signatures can be viewed in an installing org.

Permission Sets and Profile Settings in Packages

Developers can use permission sets or profile settings to grant permissions and other access settings to a package. When deciding whether to use permission sets, profile settings, or a combination of both, consider the similarities and differences.

Behavior	Permission Sets	Profile Settings
What permissions and settings are included?	• Assigned custom apps • Custom object permissions • External object permissions • Custom field permissions • Custom metadata types permissions • Custom permissions • Custom settings permissions • Custom tab visibility settings • Apex class access • Visualforce page access • External data source access • Record types	• Assigned custom apps • Assigned connected apps • Tab settings • Page layout assignments • Record type assignments • Custom field permissions • Custom metadata type permissions • Custom object permissions • Custom permissions • Custom settings permissions • External object permissions • Apex class access • Visualforce page access

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Permission sets available in: **Contact Manager**, **Professional**, **Group**, **Enterprise**, **Performance**, **Unlimited**, **Developer**, and **Database.com** Editions

Behavior	Permission Sets	Profile Settings
	Note: Although permission sets include standard tab visibility settings, these settings can't be packaged as permission set components. If a permission set includes an assigned custom app, it's possible that a subscriber can delete the app. In that case, when the package is later upgraded, the assigned custom app is removed from the permission set.	• External data source access
Can they be upgraded in managed packages?	Yes.	Profile settings are applied to existing profiles in the subscriber's org on install or upgrade. Only permissions related to new components created as part of the install or upgrade are applied.
Can subscribers edit them?	Subscribers can edit permission sets in unmanaged packages, but not in managed packages.	Yes.
Can you clone or create them?	Yes. However, if a subscriber clones a permission set or creates one that's based on a packaged permission set, it isn't updated in subsequent upgrades. Only the permission sets included in a package are upgraded.	Yes. Subscribers can clone any profile that includes permissions and settings related to packaged components.
Do they include standard object permissions?	No. Also, you can't include object permissions for a custom object in a master-detail relationship where the master is a standard object.	No.
Do they include user permissions?	No.	No.
Are they included in the installation wizard?	No. Subscribers must assign permission sets after installation.	Yes. Profile settings are applied to existing profiles in the subscriber's org on install or upgrade. Only permissions related to new components created as part of the install or upgrade are applied.
What are the user license requirements?	A permission set is only installed if the subscriber org has at least one user license that matches the permission set. For example, permission sets with the Salesforce Platform user license aren't installed in an org that has no Salesforce Platform user	None. In a subscriber org, the installation overrides the profile settings, not their user licenses.

Behavior	Permission Sets	Profile Settings
	licenses. If a subscriber later acquires a license, the subscriber must reinstall the package to get the permission sets associated with the newly acquired license. Permission sets with no user license are always installed. If you assign a permission set that doesn't include a user license, the user's existing license must allow its enabled settings and permissions. Otherwise, the assignment fails.	
How are they assigned to users?	Subscribers must assign packaged permission sets after installing the package.	Profile settings are applied to existing profiles.

Best Practices

- Use permission sets in addition to packaged profiles so your subscribers can easily add new permissions for existing app users.
- If users need access to apps, standard tabs, page layouts, and record types, don't use permission sets as the sole permission-granting model for your app.
- Create packaged permission sets that grant access to the custom components in a package, but not standard Salesforce components.

View Package Details

From Setup, enter *Packages* in the Quick Find box, then select **Packages**. Click the name of a package to view its details, including added components, whether it's a managed package, whether the package has been uploaded, and so on.

The detail page has the following sections.

- Package Detail
- Components tab
- Versions tab
- Patch Organizations tab

From the Package Detail page, you can:

- Click **Edit** to change the package name, custom link that displays when users click **Configure**, or description.
- Click **Delete** to delete the package. This does not delete the components contained in the package but the components will no longer be bundled together within this package.
- Click **Upload** to upload the package. You will receive an email when the upload is complete. For more information, see Manage Versions on page 90.
- Optionally, you can enable, disable, or change the dynamic Apex and API access that components in the package have to standard objects in the installing organization by using the links next to API Access.

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create packages:
- Create AppExchange Packages

Viewing Package Details

For package developers, the package detail section displays the following package attributes (in alphabetical order):

Attribute	Description
API Access	The type of access that the API and dynamic Apex that package components have. The default setting is **Unrestricted**, which means that all package components that access the API have the same access as the user who is logged in. Click **Enable Restrictions** or **Disable Restrictions** to change the API and dynamic Apex access permissions for a package.
Created By	The name of the developer that created this package, including the date and time.
Description	A detailed description of the package.
Language	The language used for the labels on components. The default value is your user language.
Last Modified By	The name of the last user to modify this package, including the date and time.
Notify on Apex Error	The username of the person who should receive an email notification if an exception occurs in Apex that is not caught by the code. If you do not specify a username, all uncaught exceptions generate an email notification that is sent to Salesforce. This is only available for managed packages. Note: Apex can only be packaged from Developer, Enterprise, Unlimited, and Performance Edition organizations.
Package Name	The name of the package, given by the publisher.
Post Install Script	The Apex code that runs after this package is installed or upgraded. For more information, see Running Apex on Package Install/Upgrade.
Type	Indicates whether this is a managed or unmanaged package.
Uninstall Script	The Apex code that runs after this package is uninstalled. For more information, see Running Apex on Package Uninstall.

Viewing Package Components

For package developers, the Components tab lists every package component contained in the package, including the name and type of each component.

Click **Add** to add components to the package on page 13.

Note:

- Some related components are automatically included in the package even if they don't display in the Package Components list. For example, when you add a custom object to a package, its custom fields, page layouts, and relationships with standard objects are automatically included. For a complete list of components Salesforce automatically includes, see the *ISVforce Guide*.
- When you package a joined report, each block is included in the package. Although the blocks appear in the package as reports, when you click on a block, an error message indicates that you have "insufficient privileges" to view the report. This is expected behavior. Instead, click the name of the joined report to run it.

Click **View Dependencies** to review a list of components that rely on other components, permissions, or preferences within the package. An entity may include such things as an s-control, a standard or custom field, or an organization-wide setting like multicurrency. Your package cannot be installed unless the installer has the listed components enabled or installed. For more information on dependencies, see Understanding Dependencies on page 58. Click **Back to Package** to return to the Package detail page.

Click **View Deleted Components** to see which components were deleted from the package across all of its versions.

The following component information is displayed (in alphabetical order):

Attribute	Description
`Action`	Lists the actions you can perform on the component. The only choice is **Remove**, which removes the component from an unreleased package.
`Available in Versions`	Displays the version number of the package in which a component exists.
`Included By`	This column lists how a component was included in a package. If the component was automatically included because it's referenced by another component, the column lists the name of the referencing component. If the component was added by a developer, this column lists `User Selected`. If the component was added in a prior version of this package, this column lists `Previously Released`.
`Name`	Displays the name of the component.
`Owned By`	If the component was added from a different installed package, this column lists the name of that package.
`Parent Object`	Displays the name of the parent object a component is associated with. For example, a custom object is the parent of a custom field.
`Type`	Displays the type of the component.

Viewing Version History

For package developers, the Versions tab lists all the previous uploads of a package.

Click **Push Upgrades** to automatically upgrade subscribers to a specific version.

Click the Version Number of any listed uploads to manage that upload. For more information, see Manage Versions on page 90.

Note: **Push Upgrades** is only available for patch updates. Registered Salesforce ISVs may request this feature through the Salesforce partner portal.

The versions table displays the following package attributes (in alphabetical order):

Attribute	Description
`Action`	Lists the actions you can perform on the package. The possible actions are: • **Deprecate**: Deprecates a package version. Warning: Users will no longer be able to download or install this package. However, existing installations will continue to work. • **Undeprecate**: Enables a package version to be installed by users once again.
`Status`	The status of the package. The possible statuses are: • Released: The package is Managed - Released. • Beta: The package is Managed - Beta. • Deprecated: The package version is deprecated.
Version Name	The version name for this package version. The version name is the marketing name for a specific release of a package. It is more descriptive than the `Version Number`.
Version Number	The version number for the latest installed package version. The format is *`majorNumber.minorNumber.patchNumber`*, such as 2.1.3. The version number represents a release of a package. The `Version Name` is a more descriptive name for the release. The `patchNumber` is generated only when you create a patch. If there is no `patchNumber`, it is assumed to be zero (0).

Viewing Patch Development Organizations

Every patch is developed in a *patch development organization*, which is the organization where patch versions are developed, maintained, and uploaded. To start developing a patch, you need to create a patch development organization. To do this, see Create and Upload Patches on page 90. Patch development organizations are necessary to permit developers to make changes to existing components without causing incompatibilities between existing subscriber installations. Click **New** to create a new patch for this package.

The Patch Organizations table lists all the patch development organizations created. It lists the following attributes (in alphabetical order):

Attribute	Description
`Action`	Lists the actions you can perform on a patch development organization. The possible actions are: • **Login**: Log into your patch development organization. • **Reset**: Emails a new temporary password for your patch development organization.
`Administrator Username`	The login associated with the patch organization.

Attribute	Description
`Patching Major Release`	The package version number that you are patching.

SEE ALSO:

- Create a Package
- Prepare Your Apps for Distribution
- Manage Packages

Determining Your Development Process

All packages are unmanaged until you convert them to managed packages. This requires managed packages created in a Developer Edition organization. You may prefer developing managed packages because you can beta test them before a release and offer upgrades for them.

Before creating a package, determine the development process you aim to take so that you can choose the most appropriate type of package for your process:

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Developing Unmanaged Packages

- Design your app. See the Lightning Platform Quick Reference for Developing Packages.
- Package and upload your app.

Developing Managed Packages

- Design your app. See the Lightning Platform Quick Reference for Developing Packages.
- Package and upload a beta version of your app.
- Gather feedback from your beta testers and make the appropriate fixes to your app.
- Package and upload your final version of the app.

IN THIS SECTION:

- Planning the Release of Managed Packages
- Delete Components from First-Generation Managed Packages
- Viewing Unused Components in a Package
- Modifying Custom Fields after a Package is Released
- Configuring Default Package Versions for API Calls
- About API and Dynamic Apex Access in Packages
- Manage API and Dynamic Apex Access in Packages
- Generating an Enterprise WSDL with Managed Packages
- Understanding Dependencies

Environment Hub

The Environment Hub lets you connect, create, view, and log in to Salesforce orgs from one location. If your company has multiple environments for development, testing, and trials, the Environment Hub lets you streamline your approach to org management.

SEE ALSO:

Planning the Release of Managed Packages

Manage Packages

Create and Upload Patches

Planning the Release of Managed Packages

Releasing an AppExchange package is similar to releasing any other program in software development. You may want to roll it out in iterations to ensure each component functions as planned. You may even have beta testers who have offered to install an early version of your package and provide feedback.

Once you release a package by publishing it on AppExchange, anyone can install it. So, plan your release carefully. Review the states defined below to familiarize yourself with the release process. Salesforce automatically applies the appropriate state to your package and components depending on the upload settings you choose and where it is in the release process.

State	Description
Unmanaged	The package has not been converted into a managed package or the component has not been added to a managed package. Note that a component that is "Managed - Beta" can become "Unmanaged" if it is removed from a managed package. All packages are unmanaged unless otherwise indicated by one of the managed icons below.
Managed - Beta	The package or component was created in the current Salesforce organization and is managed, but it is not released because of one of these reasons: • It has not been uploaded. • It has been uploaded with `Managed - Beta` option selected. This option prevents it from being published, publicly available on AppExchange. The developer can still edit any component but the installer may not be able to depending on which components were packaged. Note: Don't install a Managed - Beta package over a Managed - Released package. If you do, the package is no longer upgradeable and your only option is to uninstall and reinstall it.
Managed - Released	The package or component was created in the current Salesforce organization and is managed. It is also uploaded with the `Managed - Released` option selected, indicating that it can be published on AppExchange and is publicly available. Note that once you have moved a package to this state, some properties of the components are no longer editable for both the developer and installer. This type of release is considered a major release.
Patch	If you need to provide a minor upgrade to a managed package, consider creating a patch instead of a new major release. A patch enables a developer to change the functionality of existing components in a managed package. Subscribers experience no visible changes to the package.

State	Description
	This type of release is considered a patch release.
Managed - Installed	The package or component was installed from another Salesforce organization but is managed.

A developer can refine the functionality in a managed package over time, uploading and releasing new versions as the requirements evolve. This might involve redesigning some of the components in the managed package. Developers can delete some, but not all, types of components in a Managed - Released package when upgrading it.

SEE ALSO:

Manage Packages

Determining Your Development Process

Delete Components from First-Generation Managed Packages

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

Available in: **Developer** Edition

After you've uploaded a Managed - Released first-generation managed package, you may find that a component needs to be deleted from your packaging org. One of the following situations may occur:

- The component, once added to a package, can't be deleted.
- The component can be deleted, but can only be undeleted from the Deleted Package Components page.
- The component can be deleted, but can be undeleted from either the Deleted Package Components page or through the Recycle Bin

After a package is uploaded with a component marked for deletion, the component is deleted forever.

Warning: When a component is deleted, its **Name** remains within Salesforce, and you can't create a new component that uses the deleted component's name. The Deleted Package Components page lists the names that can no longer be used.

To access the Deleted Package Components page, from Setup, enter `Packages` in the `Quick Find` box, then select **Packages**. Select the package that the component was uploaded to, and then click **View Deleted Components**. You can retrieve components from the Recycle Bin and Deleted Package Components page any time *before* uploading a new version of the package. To do this, click **Undelete** next to the component.

You can retrieve these types of components.

- Apex classes and triggers that don't have `global` access.
- Visualforce components with `public` access. (If the ability to remove components has been enabled for your packaging org then these Visualforce components can't be undeleted. As a result, they don't show up in the Recycle Bin or the Deleted Package Components page after they have been deleted.)
- Protected components, including:
  - Custom labels
  - Custom links (for Home page only)
  - Custom metadata types
  - Custom permissions

  - Custom settings
  - Workflow alerts
  - Workflow field updates
  - Workflow outbound messages
  - Workflow tasks
  - Workflow flow triggers

    The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

- Data components, such as Documents, Dashboards, and Reports. These components are the only types that can also be undeleted from the Recycle Bin.

You can retrieve components from the Recycle Bin and Deleted Package Components page any time *before* uploading a new version of the package. To do this, click **Undelete** next to the component.

The Deleted Components displays the following information (in alphabetical order):

Attribute	Description
`Action`	If the Managed - Released package hasn't been uploaded with the component deleted, this contains an **Undelete** link that allows you to retrieve the component.
`Available in Versions`	Displays the version number of the package in which a component exists.
`Name`	Displays the name of the component.
`Parent Object`	Displays the name of the parent object a component is associated with. For example, a custom object is the parent of a custom field.
`Type`	Displays the type of the component.

Viewing Unused Components in a Package

This table shows components no longer being used in the current version of a package. Any component shown here that's part of a managed package is safe to delete unless you've used it in custom integrations. After you've deleted an unused component, it appears in this list for 15 days. During that time, you can either undelete it to restore the component and all data stored in it, or delete the component permanently. Note that when you undelete a custom field, some properties on the field will be lost or changed. After 15 days, the field and its data are permanently deleted.

Note: Before deleting a custom field, you can keep a record of its data. From Setup, enter *`Data Export`* in the `Quick Find` box, then select **Data Export**.

The following component information is displayed (in alphabetical order):

EDITIONS

Available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Attribute	Description
`Action`	Can be one of two options: • **Undelete** • **Delete**
`Name`	Displays the name of the component.
`Parent Object`	Displays the name of the parent object a component is associated with. For example, a custom object is the parent of a custom field.
`Type`	Displays the type of the component.

Modifying Custom Fields after a Package is Released

The following changes are allowed to custom fields in a package, after it is released.

- The length of a text field can be increased or decreased.
- The number of digits to the left or right of the decimal point in a number field can be increased or decreased.
- A required field can be made non-required and vice-versa. If a default value was required for a field, that restriction can be removed and vice-versa.

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

Available in: **Developer** Edition

Configuring Default Package Versions for API Calls

A package version is a number that identifies the set of components uploaded in a package. The version number has the format *majorNumber.minorNumber.patchNumber* (for example, 2.1.3). The major and minor numbers increase to a chosen value during every major release. The *patchNumber* is generated and updated only for a patch release. Publishers can use package versions to evolve the components in their managed packages gracefully by releasing subsequent package versions without breaking existing customer integrations using the package.

Default package versions for API calls provide fallback settings if package versions are not provided by an API call. Many API clients do not include package version information, so the default settings maintain existing behavior for these clients.

You can specify the default package versions for enterprise API and partner API calls. The enterprise WSDL is for customers who want to build an integration with their Salesforce organization only. It is strongly typed, which means that calls operate on objects and fields with specific data types, such as `int` and `string`. The partner WSDL is for customers, partners, and ISVs who want to build an integration that can work across multiple Salesforce organizations, regardless of their custom objects or fields. It is loosely typed, which means that calls operate on name-value pairs of field names and values instead of specific data types.

EDITIONS

Available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer**, Editions

USER PERMISSIONS

To configure default package versions for API calls:

- Customize Application

You must associate the enterprise WSDL with specific package versions to maintain existing behavior for clients. There are options for setting the package version bindings for an API call from client applications using either the enterprise or partner WSDL. The package version information for API calls issued from a client application based on the enterprise WSDL is determined by the first match in the following settings.

1. The PackageVersionHeader SOAP header.

2. The SOAP endpoint contains a URL with a format of *serverName*/services/Soap/c/*api_version*/*ID* where *api_version* is the version of the API, such as , and *ID* encodes your package version selections when the enterprise WSDL was generated.
3. The default enterprise package version settings.

The partner WSDL is more flexible as it is used for integration with multiple organizations. If you choose the Not Specified option for a package version when configuring the default partner package versions, the behavior is defined by the latest installed package version. This means that behavior of package components, such as an Apex trigger, could change when a package is upgraded and that change would immediately impact the integration. Subscribers may want to select a specific version for an installed package for all partner API calls from client applications to ensure that subsequent installations of package versions do not affect their existing integrations.

The package version information for partner API calls is determined by the first match in the following settings.

1. The PackageVersionHeader SOAP header.
2. An API call from a Visualforce page uses the package versions set for the Visualforce page.
3. The default partner package version settings.

To configure default package versions for API calls:

1. From Setup, enter *API* in the Quick Find box, then select **API**.
2. Click **Configure Enterprise Package Version Settings** or **Configure Partner Package Version Settings**. These links are only available if you have at least one managed package installed in your organization.
3. Select a Package Version for each of your installed managed packages. If you are unsure which package version to select, you should leave the default selection.
4. Click **Save**.

Note: Installing a new version of a package in your organization does not affect the current default settings.

About API and Dynamic Apex Access in Packages

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

Available in: **Contact Manager**, **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Apex Package components have access via dynamic Apex and the API to standard and custom objects in the organization where they are installed.

API Access is a package setting that controls the dynamic Apex and API access that s-controls and other package components have to standard and custom objects. The setting displays for both the developer and installer on the package detail page. With this setting:

- The developer of an AppExchange package can restrict API access for a package before uploading it to Salesforce AppExchange. Once restricted, the package components receive Apex and API sessions that are restricted to the custom objects in the package. The developer can also enable access to specific standard objects, and any custom objects in other packages that this package depends on.
- The installer of a package can accept or reject package access privileges when installing the package to his or her organization.
- After installation, an administrator can change Apex and API access for a package at any time. The installer can also enable access on additional objects such as custom objects created in the installer's organization or objects installed by unrelated packages.

There are two possible options for the API Access setting:

- The default Unrestricted, which gives the package components the same API access to standard objects as the user who is logged in when the component sends a request to the API. Apex runs in system mode. Unrestricted access gives Apex read access to all standard and custom objects.

- `Restricted`, which allows the administrator to select which standard objects the components in the package can access. Further, the components in restricted packages can only access custom objects in the current package if the user has the object permissions that provide access to them.

Considerations for API and Dynamic Apex Access in Packages

By default, dynamic Apex can only access the components with which the code is packaged. To provide access to standard objects not included in the package, the developer must set the `API Access`.

1. From Setup, enter *`Packages`* in the `Quick Find` box, then select **Packages**.
2. Select the package that contains a dynamic Apex that needs access to standard objects in the installing organization.
3. In the Package Detail related list, click **Enable Restrictions** or `Restricted`, whichever is available.
4. Set the access level (Read, Create, Edit, Delete) for the standard objects that the dynamic Apex can access.
5. Click **Save**.

Choosing `Restricted` for the `API Access` setting in a package affects the following:

- API access in a package overrides the following user permissions:
  - Author Apex
  - Customize Application
  - Edit HTML Templates
  - Edit Read Only Fields
  - Manage Billing
  - Manage Call Centers
  - Manage Categories
  - Manage Custom Report Types
  - Manage Dashboards
  - Manage Letterheads
  - Manage Package Licenses
  - Manage Public Documents
  - Manage Public List Views
  - Manage Public Reports
  - Manage Public Templates
  - Manage Users
  - Transfer Record
  - Use Team Reassignment Wizards
  - View Setup and Configuration
  - Weekly Export Data
- If `Read`, `Create`, `Edit`, and `Delete` access are not selected in the API access setting for objects, users do not have access to those objects from the package components, even if the user has the "Modify All Data" and "View All Data" permissions.
- A package with `Restricted` API access can't create new users.
- Salesforce denies access to Web service and `executeanonymous` requests from an AppExchange package that has `Restricted` access.

The following considerations also apply to API access in packages:

- Workflow rules and Apex triggers fire regardless of API access in a package.
- If a component is in more than one package in an organization, API access is unrestricted for that component in all packages in the organization regardless of the access setting.
- If Salesforce introduces a new standard object after you select restricted access for a package, access to the new standard object is not granted by default. You must modify the restricted access setting to include the new standard object.
- When you upgrade a package, changes to the API access are ignored even if the developer specified them. This ensures that the administrator installing the upgrade has full control. Installers should carefully examine the changes in package access in each upgrade during installation and note all acceptable changes. Then, because those changes are ignored, the administrator should manually apply any acceptable changes after installing an upgrade.
- S-controls are served by Salesforce and rendered inline in Salesforce. Because of this tight integration, there are several means by which an s-control in an installed package could escalate its privileges to the user's full privileges. In order to protect the security of organizations that install packages, s-controls have the following limitations:
  - For packages you are developing (that is, not installed from AppExchange), you can only add s-controls to packages with the default `Unrestricted` API access. Once a package has an s-control, you cannot enable `Restricted` API access.
  - For packages you have installed, you can enable access restrictions even if the package contains s-controls. However, access restrictions provide only limited protection for s-controls. Salesforce recommends that you understand the JavaScript in an s-control before relying on access restriction for s-control security.
  - If an installed package has `Restricted` API access, upgrades will be successful only if the upgraded version does not contain any s-controls. If s-controls are present in the upgraded version, you must change the currently installed package to `Unrestricted` API access.

SEE ALSO:

Install a Package

Configuring Installed Packages

Manage API and Dynamic Apex Access in Packages

`API Access` is a package setting that controls the dynamic Apex and API access that s-controls and other package components have to standard and custom objects. The setting displays for both the developer and installer on the package detail page. With this setting:

- The developer of an AppExchange package can restrict API access for a package before uploading it to Salesforce AppExchange. Once restricted, the package components receive Apex and API sessions that are restricted to the custom objects in the package. The developer can also enable access to specific standard objects, and any custom objects in other packages that this package depends on.
- The installer of a package can accept or reject package access privileges when installing the package to his or her organization.
- After installation, an administrator can change Apex and API access for a package at any time. The installer can also enable access on additional objects such as custom objects created in the installer's organization or objects installed by unrelated packages.

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

Available in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To edit API and dynamic Apex access for a package you have created or installed:

- Create AppExchange packages

To accept or reject package API and dynamic Apex access for a package as part of installation:

- Download AppExchange packages

Setting API and Dynamic Apex Access in Packages

To change package access privileges in a package you or someone in your organization has created:

1. From Setup, enter *Packages* in the `Quick Find` box, then select **Packages**.
2. Select a package.
3. The `API Access` field displays the current setting, `Restricted` or `Unrestricted`, and a link to either **Enable Restrictions** or **Disable Restrictions**. If `Read`, `Create`, `Edit`, and `Delete` access are not selected in the API access setting for objects, users do not have access to those objects from the package components, even if the user has the "Modify All Data" and "View All Data" permissions.

   Use the `API Access` field to:

   Enable Restrictions
   This option is available only if the current setting is `Unrestricted`. Select this option if you want to specify the dynamic Apex and API access that package components have to standard objects in the installer's organization. When you select this option, the Extended Object Permissions list is displayed. Select the `Read`, `Create`, `Edit`, or `Delete` checkboxes to enable access for each object in the list. This selection is disabled in some situations. Click **Save** when finished. For more information about choosing the `Restricted` option, including information about when it is disabled, see Considerations for API and Dynamic Apex Access in Packages on page 54.

   Disable Restrictions
   This option is available only if the current setting is `Restricted`. Select this option if you do not want to restrict the Apex and API access privileges that the components in the package have to standard and custom objects. This option gives all the components in the package the same API access as the user who is logged in. For example, if a user can access accounts, an Apex class in the package that accesses accounts would succeed when triggered by that user.

   Restricted
   Click this link if you have already restricted API access and wish to edit the restrictions.

Accepting or Rejecting API and Dynamic Apex Access Privileges During Installation

To accept or reject the API and dynamic Apex access privileges for a package you are installing:

- Start the installation process on Salesforce AppExchange.

- In **Approve API Access**, either accept by clicking **Next**, or reject by clicking **Cancel**. Complete the installation steps if you have not canceled.

Changing API and Dynamic Apex Access Privileges After Installation

To edit the package API and dynamic Apex access privileges after you have installed a package:

1. From Setup, enter *Installed Packages* in the Quick Find box, then select **Installed Packages**.
2. Click the name of the package you wish to edit.
3. The API Access field displays the current setting, Restricted or Unrestricted, and a link to either **Enable Restrictions** or **Disable Restrictions**. If Read, Create, Edit, and Delete access are not selected in the API access setting for objects, users do not have access to those objects from the package components, even if the user has the "Modify All Data" and "View All Data" permissions.

   Use the API Access field to:

   Enable Restrictions
   This option is available only if the current setting is Unrestricted. Select this option if you want to specify the dynamic Apex and API access that package components have to standard objects in the installer's organization. When you select this option, the Extended Object Permissions list is displayed. Select the Read, Create, Edit, or Delete checkboxes to enable access for each object in the list. This selection is disabled in some situations. Click **Save** when finished. For more information about choosing the Restricted option, including information about when it is disabled, see Considerations for API and Dynamic Apex Access in Packages on page 54.

   Disable Restrictions
   This option is available only if the current setting is Restricted. Select this option if you do not want to restrict the Apex and API access privileges that the components in the package have to standard and custom objects. This option gives all the components in the package the same API access as the user who is logged in. For example, if a user can access accounts, an Apex class in the package that accesses accounts would succeed when triggered by that user.

   Restricted
   Click this link if you have already restricted API access and wish to edit the restrictions.

Generating an Enterprise WSDL with Managed Packages

If you are downloading an enterprise WSDL and you have managed packages installed in your organization, you need to take an extra step to select the version of each installed package to include in the generated WSDL. The enterprise WSDL is strongly typed, which means that it contains objects and fields with specific data types, such as int and string.

A package version is a number that identifies the set of components uploaded in a package. The version number has the format *majorNumber.minorNumber.patchNumber* (for example, 2.1.3). The major and minor numbers increase to a chosen value during every major release. The *patchNumber* is generated and updated only for a patch release. Publishers can use package versions to evolve the components in their managed packages gracefully by releasing subsequent package versions without breaking existing customer integrations using the package. A subscriber can select a package version for each installed managed package to allow their API client to continue to function with specific, known behavior even when they install subsequent versions of a package. Each package version can have variations in the composition of its objects and fields, so you must select a specific version when you generate the strongly typed WSDL.

To download an enterprise WSDL when you have managed packages installed:

EDITIONS

Available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer**, Editions

USER PERMISSIONS

To download a WSDL:
- Customize Application

1. From Setup, enter *API* in the `Quick Find` box, then select **API**.
2. Click **Generate Enterprise WSDL**.
3. Select the `Package Version` for each of your installed managed packages. If you are unsure which package version to select, you should leave the default, which is the latest package version.
4. Click **Generate**.
5. Use the **File** menu in your browser to save the WSDL to your computer.
6. On your computer, import the local copy of the WSDL document into your development environment.

Note the following in your generated enterprise WSDL:

- Each of your managed package version selections is included in a comment at the top of the WSDL.
- The generated WSDL contains the objects and fields in your organization, including those available in the selected versions of each installed package. If a field or object is added in a later package version, you must generate the enterprise WSDL with that package version to work with the object or field in your API integration.
- The SOAP endpoint at the end of the WSDL contains a URL with a format of *serverName*`/services/Soap/c/`*api_version*`/`*ID* where *api_version* is the version of the API, such as , and *ID* encodes your package version selections when you communicate with Salesforce.

You can also select the default package versions for the enterprise WSDL without downloading a WSDL from the API page in Setup. Default package versions for API calls provide fallback settings if package versions are not provided by an API call. Many API clients do not include package version information, so the default settings maintain existing behavior for these clients.

Understanding Dependencies

Package dependencies are created when one component references another component, permission, or preference that is required for the component to be valid. Lightning Platform tracks certain dependencies, including:

- Organizational dependencies, such as whether multicurrency or campaigns are enabled
- Component-specific dependencies, such as whether particular record types or divisions exist
- References to both standard and custom objects or fields

Packages, Apex classes, Apex triggers, Visualforce components, and Visualforce pages can have dependencies on components within an organization. These dependencies are recorded on the Show Dependencies page.

Dependencies are important for packaging because any dependency in a component of a package is considered a dependency of the package as a whole.

Note: An installer's organization must meet all dependency requirements listed on the Show Dependencies page or else the installation will fail. For example, the installer's organization must have divisions enabled to install a package that references divisions.

Dependencies are important for Apex classes or triggers. Any component on which a class or trigger depends must be included with the class or trigger when the code is deployed or packaged.

In addition to dependencies, the *operational scope* is also displayed on the Show Dependencies page. The operational scope is a table that lists any data manipulation language (DML) operations (such as `insert` or `merge`) that Apex executes on a specified object. The operational scope can be used when installing an application to determine the full extent of the application's database operations.

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

AppExchange packages and Visualforce are available in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Apex available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To upload packages:
- Upload AppExchange Packages

To view Visualforce dependencies:
- Developer Mode

To view the dependencies and operational scope for a package, Apex class, Apex trigger, or Visualforce page:

1. Navigate to the appropriate component from Setup:
   - For packages, enter *Packages* in the Quick Find box, then select **Packages**.
   - For Apex classes, enter *Apex Classes* in the Quick Find box, then select **Apex Classes**.
   - For Apex triggers, from the management settings for the appropriate object, go to Triggers.
   - For Visualforce pages, enter *Visualforce Pages* in the Quick Find box, then select **Visualforce Pages**.
2. Select the name of the component.
3. Click **View Dependencies** for a package, or **Show Dependencies** for all other components, to see a list of objects that depend upon the selected component.

If a list of dependent objects displays, click **Fields** to access the field-level detail of the operational scope. The field-level detail includes information, such as whether Apex updates a field. For more information, see Field Operational Scope.

Packages, Apex code, and Visualforce pages can depend many components, including but not limited to:

- Custom field definitions
- Validation formulas
- Reports
- Record types
- Apex
- Visualforce pages and components

For example, if a Visualforce page includes a reference to a multicurrency field, such as `{!contract.ISO_code}`, that Visualforce page has a dependency on multicurrency. If a package contains this Visualforce page, it also has a dependency on multicurrency. Any organization that wants to install this package must have multicurrency enabled.

SEE ALSO:

Prepare Your Apps for Distribution
Manage Versions
Publish Upgrades to Managed Packages
Publishing Extensions to Managed Packages
Field Operational Scope

Environment Hub

The Environment Hub lets you connect, create, view, and log in to Salesforce orgs from one location. If your company has multiple environments for development, testing, and trials, the Environment Hub lets you streamline your approach to org management.

From the Environment Hub, you can:

- Connect existing orgs to the hub with automatic discovery of related orgs.
- Create standard and partner edition orgs for development, testing, and trials.
- View and filter hub members according to criteria that you choose, like edition, creation date, instance, origin, and SSO status.
- Create single sign-on (SSO) user mappings for easy login access to hub members.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions

Each hub member org corresponds to an EnvironmentHubMember object. EnvironmentHubMember is a standard object, similar to Accounts or Contacts, so you can use the platform to extend or modify the Environment Hub programmatically. For example, you can create custom fields, set up workflow rules, or define user mappings and enable SSO using the API for any hub member org.

IN THIS SECTION:

Get Started with the Environment Hub

Configure the Environment Hub so that users at your company can access the app to create and manage member orgs. Then enable My Domain so that you can connect existing orgs to the hub and create SSO user mappings.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions

IN THIS SECTION:

Configure the Environment Hub

Enable the Environment Hub in your org, and then configure it to give other users access.

1. Contact Salesforce to enable the Environment Hub in your org. If you're an ISV partner, you can skip this step. The Environment Hub is already installed in your Partner Business Org.
2. Log in to the org where the Environment Hub is enabled, and then go to Setup.
3. Assign users access to features in the Environment Hub.
   a. From Setup, enter *Profiles* in the Quick Find box, then select **Profiles**.

USER PERMISSIONS

To set up and configure the Environment Hub:
- Manage Environment Hub

b. Create a profile, or edit an existing one.

c. Edit the profile's settings.

Profile Section	Environment Hub Settings
Custom App Settings	Enable the Environment Hub custom app to make it available in the App Launcher in Lightning Experience or App Menu in Salesforce Classic.
Connected App Access	Unless advised by Salesforce, don't adjust settings in this section of the profile.
Service Provider Access	If you enable single sign-on (SSO) in a member org, new entries appear in this section of the profile. Entries appear in the format *Service Provider [Organization ID]*, where *Organization ID* is the ID of the member org. Users who don't have access to the service provider sometimes see this message when attempting to log in via SSO: "User '[UserID]' does not have access to sp '[Service Provider ID]'." When configuring the Environment Hub in a new org, this section is empty.
Administrative Permissions	Enable "Manage Environment Hub" to allow users to: • Create orgs for development, testing, and trials. • Configure SSO for member orgs.
General User Permissions	Enable "Connect Organization to Environment Hub" to allow users to connect existing orgs to the Environment Hub.
Standard Object Permissions	Grant object permissions based on the level of access required by the Environment Hub user. Hub Members object: • "Read"—View existing Hub Member records. • "Create"—This permission has no impact on the ability to create Hub Member records. That's because record creation is handled either by connecting an existing org or creating an org from the Environment Hub. • "Edit"—Edit fields on existing Hub Member records. • "Delete"—Disconnect an org from the Environment Hub and delete its corresponding Hub Member record and Service Provider record (if SSO was enabled for the member). • "View All"—Read all Hub Member records, regardless of who created them. • "Modify All"—Read, edit, and delete all Hub Member records, regardless of who created them. Hub Invitations object:

Profile Section	Environment Hub Settings
	• If you enable the "Connect Organization to Environment Hub" permission, enable "Create", "Read", "Update, and "Delete" for Hub Invitations. Signup Request object: • If you enable the "Manage Environment Hub" permission, enable "Create" and "Read" for Signup Requests to allow users to create orgs. Optionally, enable "Delete" to allow users to remove orgs from the hub.

d. Select **Save**.

Enable My Domain for the Environment Hub

My Domain is required to connect existing Salesforce orgs to the Environment Hub and create SSO user mappings. Enable My Domain in the org where the Environment Hub is installed.

With My Domain, you specify a customer-specific name to include in your Salesforce org URLs and register it with Salesforce domain registries worldwide. You can also customize your login page and better manage user login and authentication.

1. Set Up My Domain. Choose or change your My Domain name, then update your org to use your new URLs.

   Note: Production orgs created in Winter '21 and later have a My Domain by default. If you don't like your org's My Domain name, you can change it.

2. Configure My Domain Settings. Determine the user experience when logging into your Salesforce org via your My Domain. Manage user logins and authentication methods and customize your login page with your brand.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Group**, **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To set up My Domain:
- Customize Application

To define a My Domain name:
- Customize Application

  AND

  Modify All Data

Manage Orgs in the Environment Hub

You can manage all your existing Salesforce orgs from one location by connecting them to the Environment Hub. You can also create orgs using Salesforce templates for development, testing, and trial purposes.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions

IN THIS SECTION:

Connect an Org to the Environment Hub

You can connect existing Salesforce orgs to the Environment Hub, allowing you to manage all your development, test, and trial environments (except scratch orgs) from one location. When you connect an org to the hub, related orgs are automatically discovered so you don't have to manually connect them.

Create an Org from the Environment Hub

You can create orgs from the Environment Hub for development, testing, and trial purposes. If you're an ISV partner, you can also create partner edition orgs with increased limits, more storage, and other customizations to support app development. When you create an org from the Environment Hub, it becomes a hub member and its default language is set by the user's locale.

Connect an Org to the Environment Hub

You can connect existing Salesforce orgs to the Environment Hub, allowing you to manage all your development, test, and trial environments (except scratch orgs) from one location. When you connect an org to the hub, related orgs are automatically discovered so you don't have to manually connect them.

USER PERMISSIONS

To connect or disconnect an org to or from the Environment Hub:

- Connect Organization to Environment Hub

The following types of related orgs are automatically discovered.

- For any organization, all sandbox orgs created from it
- For a release org, all its related patch orgs
- For a Trialforce Management Org, all Trialforce Source Orgs created from it
- For an org with the License Management App (LMA) installed, any release org with a managed package registered in the LMA

Note: You can't connect a sandbox org to the Environment Hub directly. If you want to connect a sandbox, first connect the org used to create the sandbox to the Environment Hub. Then, refresh the sandbox org. The refresh automatically adds it as a hub member.

1. Log in to the Environment Hub, and then select **Connect Org**.
2. Enter the admin username for the org that you want to connect and, optionally, a short description. A description makes it easier to find the org later, especially if your hub has many members.
3. By default, single sign-on (SSO) is enabled for the org you connected. To disable SSO, deselect **Auto-enable SSO for this org**.
4. Select **Connect Org** again.
5. In the pop-up window, enter the org's admin username and password. If you don't see the pop-up, temporarily disable your browser's ad blocking software and try again.
6. Select **Log In**, and then select **Allow**.

To disconnect an org, locate the listing for the org, and select **Remove** from the dropdown menu on the far right.

Orgs removed from the Environment Hub aren't deleted, so you can still access the org after you remove it.

Create an Org from the Environment Hub

You can create orgs from the Environment Hub for development, testing, and trial purposes. If you're an ISV partner, you can also create partner edition orgs with increased limits, more storage, and other customizations to support app development. When you create an org from the Environment Hub, it becomes a hub member and its default language is set by the user's locale.

Note: You can create up to 20 member orgs per day. To create more orgs, log a case in the Partner Community.

USER PERMISSIONS

To set up and configure the Environment Hub:
- Manage Environment Hub

1. Log in to the Environment Hub, and then select **Create Org**.
2. Choose an org purpose.

Purpose	Lets You Create:
Development	Developer Edition orgs for building and packaging apps.
Test/Demo	Trial versions of standard Salesforce orgs for testing and demos. These orgs are similar to the ones customers create at www.salesforce.com/trial. When you create a Test/Demo org, you can specify a Trialforce template if you want the org to include your customizations.
Trialforce Source Organization	Trialforce Source Organizations (TSOs) as an alternative to using a Trialforce Management Organization (TMO). Unless you need custom branding on your login page or emails, use the Environment Hub to create TSOs. If you're creating a TSO from an Environment Hub that is also a TMO, you can't set a My Domain subdomain

3. Enter the required information for the org type you selected.
4. Read the Master Subscription Agreement, and then select the checkbox.
5. Select **Create**.

When your org is ready, you receive an email confirmation, and the org appears in your list of hub members.

Single Sign-on in the Environment Hub

Developing, testing, and deploying apps means switching between multiple Salesforce environments and providing login credentials each time. Single sign-on (SSO) simplifies this process by letting an Environment Hub user log in to member orgs without reauthenticating. You can set up SSO by defining user mappings manually, using Federation IDs, or creating a formula.

The Environment Hub supports these SSO methods for matching users.

SSO Method	Description
Mapped Users	Match users in the Environment Hub to users in a member org manually. Mapped Users is the default method for SSO user mappings defined from the member detail page.
Federation ID	Match users who have the same Federation ID in both the Environment Hub and a member org.
User Name Formula	Match users in the Environment Hub and a member org according to a formula that you define.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions

If you specify multiple SSO methods, they're evaluated in this order: (1) Mapped Users, (2) Federation ID, and (3) User Name Formula. The first method that results in a match is used to log in the user, and the other methods are ignored. If a matching user can't be identified, the Environment Hub directs the user to the standard Salesforce login page.

Note: SSO doesn't work for newly added users or for user mappings defined in a sandbox org. Only add users, edit user information, or define SSO user mappings in the parent org for the sandbox.

IN THIS SECTION:

Enable SSO for a Member Org

You can enable single sign-on (SSO) to let an Environment Hub user log in to a member org without reauthenticating.

1. Log in to the Environment Hub, and then select a member org. If you don't see any member orgs, check your list view.
2. Select **Enable SSO**.
3. Confirm that you want to enable SSO for this org, and then select **Enable SSO** again.

USER PERMISSIONS

To set up and configure the Environment Hub:
- Manage Environment Hub

Define an SSO User Mapping

You can manually define a single-sign on (SSO) user mapping between a user in the Environment Hub and a user in a member org. Before you define a user mapping, enable SSO in the hub member org.

User mappings can be many-to-one but not one-to-many. In other words, you can associate multiple users in the Environment Hub to one user in a member org. For example, if you wanted members of your QA team to log in to a test org as the same user, you could define user mappings.

USER PERMISSIONS

To set up and configure the Environment Hub:
- Manage Environment Hub

1. Log in to the Environment Hub, and then select a member org. If you don't see any member orgs, check your list view.
2. Go to the Single Sign-On User Mappings related list, and then select **New SSO User Mapping**.
3. Enter the username of the user that you want to map in the member org, and then look up a user in the Environment Hub.
4. Select **Save**.

Use a Federation ID or Formula for SSO

You can match an Environment Hub user with a user in a member org using a Federation ID or a user name formula. For either method, enable SSO in the hub member org first.

1. Log in to the Environment Hub, and then select a member org. If you don't see any member orgs, check your list view.
2. Go to SSO Settings, and then choose a method.

Method	Steps
`SSO Method 2 - Federation ID`	Select the checkbox.
`SSO Method 3 - User Name Formula`	Select the checkbox, and then define a formula. For example, to match the first part of the username (the part before the "@" sign) with an explicit domain name, enter: `LEFT($User.Username, FIND("@", $User.Username)) & ("mydev.org")`

3. Select **Save**.

USER PERMISSIONS

To set up and configure the Environment Hub:
- Manage Environment Hub

Disable SSO for a Member Org

If you want Environment Hub users to reauthenticate when they log in to a member org, you can disable SSO. Disabling SSO doesn't remove the user mappings that you've defined, so you can always re-enable SSO later.

1. Log in to the Environment Hub, and then select a member org. If you don't see any member orgs, check your list view.
2. Select **Disable SSO**.
3. Confirm that you want to disable SSO for this org, and then select **Disable SSO** again.

USER PERMISSIONS

To set up and configure the Environment Hub:
- Manage Environment Hub

Environment Hub Best Practices

Follow these guidelines and best practices when you use the Environment Hub.

- If you're an admin or developer, choose the org that your team uses most frequently as your hub org. If you're an ISV partner, the Environment Hub is already installed in your Partner Business Org.
- Set up My Domain for each member org, in addition to the hub org. Because each My Domain includes a unique domain URL, it's easier to distinguish between the member orgs that you use for development, testing, and trials.

  Note: My Domain subdomains are not available for Trialforce Source Organizations created from an Environment Hub that's also a Trialforce Management Organization.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions

- Because each member org is a standard object (of type EnvironmentHubMember), you can modify its behavior or access it programmatically. For example, you can create custom fields, set up workflow rules, or define user mappings and enable single sign-on using the API for any member org.
- Decide on a strategy for enabling SSO access based on your company's security requirements. Then choose the SSO method (explicit mapping, Federation ID, or custom formula) that meets your needs.
- SSO doesn't work for newly added users or for user mappings defined in a sandbox org. Only add users, edit user information, or define SSO user mappings in the parent org for the sandbox.
- The Environment Hub connected app is for internal use only. Don't enable it for any profiles. Unless advised by Salesforce, don't delete the connected app or adjust its settings.

Environment Hub FAQ

Answers to common questions about the Environment Hub.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

IN THIS SECTION:

Can I use the Environment Hub in Lightning Experience?

Yes, both Salesforce Classic and Lightning Experience support the Environment Hub.

Where do I install the Environment Hub?

If you're an ISV partner, the Environment Hub is already installed in your Partner Business Org.

Otherwise, install the Environment Hub in an org that all your users can access, such as your CRM org. Do not install the Environment Hub in a Developer Edition org that contains your managed package. Doing so can cause problems when you upload a new package version or push an upgrade to customers.

Is My Domain required to use the Environment Hub?

No, My Domain isn't required. But if you don't set up My Domain, you can't connect existing orgs to the Environment Hub or use single sign-on to log in to member orgs. Salesforce recommends setting up My Domain when you configure the Environment Hub.

Note: My Domain subdomains are not available for Trialforce Source Organizations created from an Environment Hub that's also a Trialforce Management Organization.

Can I install the Environment Hub in more than one org?

Yes, but you must manage each Environment Hub independently. Although Salesforce recommends one Environment Hub per company, several hubs could make sense for your company. For example, if you want to keep orgs that are associated with product lines separate.

Can I enable the Environment Hub in a sandbox org?

No, you can't enable the Environment Hub in a sandbox org. Enable the Environment Hub in a production org that all your users can access.

What kinds of orgs can I create in the Environment Hub?

You can create orgs for development, testing, and trials. ISV partners can also create partner edition orgs with increased limits, more storage, and other customizations to support app development. If you're a partner but don't see partner edition orgs in the Environment Hub, log a case in the Partner Community.

Org Type	Best Used For	Expires After
Group Edition	Testing	30 days
Enterprise Edition	Testing	30 days
Professional Edition	Testing	30 days
Partner Developer Edition	Developing apps and Lightning components	Never
Partner Group Edition	Robust testing and customer demos	1 year, unless you request an extension
Partner Enterprise Edition	Robust testing and customer demos	1 year, unless you request an extension
Partner Professional Edition	Robust testing and customer demos	1 year, unless you request an extension
Trialforce Source Org	Creating Trialforce templates	1 year, unless you request an extension
Consulting Partner Edition	Customer demos	1 year, unless you request an extension

How is locale determined for the orgs I create in the Environment Hub?

Your Salesforce user locale determines the default locale of orgs that you create. For example, if your user locale is set to `English (United Kingdom)`, that is the default locale for the orgs you create. In this way, the orgs you create are already customized for the regions where they reside.

Are the orgs that I create in the Environment Hub the same as the ones I created in the Partner Portal?

Yes, the orgs are identical to the ones that you created in the Partner Portal. The Environment Hub uses the same templates, so the orgs come with the same customizations, such as higher limits and more licenses. You can also use the Environment Hub to create the same Group, Professional, and Enterprise Edition orgs that customers use. That way, you can test your app against realistic customer implementations.

Can an org be a member of multiple Environment Hubs?

No, an org can be a member of only one Environment Hub at a time. To remove an org from an Environment Hub so you can associate it with a different one:

1. Go to the Environment Hub tab.
2. Find the org, from the drop-down select **Remove**.
3. Once removed, connect the org to the desired Environment Hub:
   a. In the Environment Hub tab, click **Connect Org**.
   b. Enter the admin username for the org.
   c. Click **Connect Org**.
   d. Enter the org's password, then click **Allow** to allow the Environment Hub to access org information.

Can I disable the Environment Hub?

After you install the Environment Hub in an org, you can't disable it. However, you can hide the Environment Hub from users. Go to Setup and enter *App Menu* in to the Quick Find box, and then select **App Menu**. From the App Menu, you can choose whether to hide an app or make it visible.

Considerations for the Environment Hub in Lightning Experience

Be aware of these considerations when creating and managing orgs in the Environment Hub.

List View Limitations

You can't filter hub members by org expiration date when creating or updating list views in Lightning Experience. If you have an existing list view that includes org expiration date in its filter criteria, that list view won't work in Lightning Experience. To filter hub members by org expiration date, switch to Salesforce Classic and then use the list view.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Resolving Apex Test Failures

Package installs or upgrades may fail for not passing Apex test coverage. However, some of these failures can be ignored. For example, a developer might write an Apex test that makes assumptions about a subscriber's data.

If you're a subscriber whose installation is failing due to an Apex test, contact the developer of the package for help.

If you're a developer and an install fails due to an Apex test failure, check for the following:

EDITIONS

Available in: **Developer** Edition

- Make sure that you are staging all necessary data required for your Apex test, instead of relying on subscriber data that exists.
- If a subscriber creates a validation rule, required field, or trigger on an object referenced by your package, your test might fail if it performs DML on this object. If this object is created only for testing purposes and never at runtime, and the creation fails due to these conflicts, you might be safe to ignore the error and continue the test. Otherwise, contact the customer and determine the impact.

Running Apex on Package Install/Upgrade

App developers can specify an Apex script to run automatically after a subscriber installs or upgrades a managed package. This makes it possible to customize the package install or upgrade, based on details of the subscriber's organization. For instance, you can use the script to populate custom settings, create sample data, send an email to the installer, notify an external system, or kick off a batch operation to populate a new field across a large set of data. For simplicity, you can only specify one post install script. It must be an Apex class that is a member of the package.

The post install script is invoked after tests have been run, and is subject to default governor limits. It runs as a special system user that represents your package, so all operations performed by the script appear to be done by your package. You can access this user by using UserInfo. You will only see this user at runtime, not while running tests.

If the script fails, the install/upgrade is aborted. Any errors in the script are emailed to the user specified in the **Notify on Apex Error** field of the package. If no user is specified, the install/upgrade details will be unavailable.

The post install script has the following additional properties.

- It can initiate batch, scheduled, and future jobs.
- It can't access Session IDs.
- It can only perform callouts using an async operation. The callout occurs after the script is run and the install is complete and committed.
- It can't call another Apex class in the package if that Apex class uses the `with sharing` keyword. This keyword can prevent the package from successfully installing. See the *Apex Developer Guide* to learn more.

Note: You can't run a post install script in a new trial organization provisioned using Trialforce. The script only runs when a subscriber installs your package in an existing organization.

IN THIS SECTION:

How does a Post Install Script Work?

A post install script is an Apex class that implements the `InstallHandler` interface. This interface has a single method called `onInstall` that specifies the actions to be performed on installation.

```
global interface InstallHandler {
  void onInstall(InstallContext context)
}
```

The `onInstall` method takes a context object as its argument, which provides the following information.

- The org ID of the organization in which the installation takes place.
- The user ID of the user who initiated the installation.
- The version number of the previously installed package (specified using the `Version` class). This is always a three-part number, such as 1.2.0.
- Whether the installation is an upgrade.
- Whether the installation is a push.

The context argument is an object whose type is the `InstallContext` interface. This interface is automatically implemented by the system. The following definition of the `InstallContext` interface shows the methods you can call on the context argument.

```
global interface InstallContext {
  ID organizationId();
  ID installerId();
  Boolean isUpgrade();
  Boolean isPush();
  Version previousVersion();
}
```

Version Methods and Class

You can use the methods in the `System.Version` class to get the version of a managed package and to compare package versions. A package version is a number that identifies the set of components uploaded in a package. The version number has the format *`majorNumber.minorNumber.patchNumber`* (for example, 2.1.3). The major and minor numbers increase to a chosen value during every non-patch release. Major and minor number increases will always use a patch number of 0.

The following are instance methods for the `System.Version` class.

Method	Arguments	Return Type	Description
`compareTo`	System.Version *`version`*	Integer	Compares the current version with the specified version and returns one of the following values: • Zero if the current package version is equal to the specified package version • An Integer value greater than zero if the current package version is greater than the specified package version • An Integer value less than zero if the current package version is less than the specified package version If a two-part version is being compared to a three-part version, the patch number is ignored and the comparison is based only on the major and minor numbers.
`major`		Integer	Returns the major package version of the calling code.
`minor`		Integer	Returns the minor package version of the calling code.
`patch`		Integer	Returns the patch package version of the calling code or `null` if there is no patch version.

The `System` class contains two methods that you can use to specify conditional logic, so different package versions exhibit different behavior.

- `System.requestVersion`: Returns a two-part version that contains the major and minor version numbers of a package. Using this method, you can determine the version of an installed instance of your package from which the calling code is referencing your package. Based on the version that the calling code has, you can customize the behavior of your package code.

- `System.runAs(System.Version)`: Changes the current package version to the package version specified in the argument.

When a subscriber has installed multiple versions of your package and writes code that references Apex classes or triggers in your package, they must select the version they are referencing. You can execute different code paths in your package's Apex code based on the version setting of the calling Apex code making the reference. You can determine the calling code's package version setting by calling the `System.requestVersion` method in the package code.

SEE ALSO:

Apex Developer Guide: Version Class

Example of a Post Install Script

The following sample post install script performs these actions on package install/upgrade.

- If the previous version is null, that is, the package is being installed for the first time, the script:
  - Creates a new Account called "Newco" and verifies that it was created.
  - Creates a new instance of the custom object Survey, called "Client Satisfaction Survey".
  - Sends an email message to the subscriber confirming installation of the package.
- If the previous version is 1.0, the script creates a new instance of Survey called "Upgrading from Version 1.0".
- If the package is an upgrade, the script creates a new instance of Survey called "Sample Survey during Upgrade".
- If the upgrade is being pushed, the script creates a new instance of Survey called "Sample Survey during Push".

```
global class PostInstallClass implements InstallHandler {
  global void onInstall(InstallContext context) {
    if(context.previousVersion() == null) {
      Account a = new Account(name='Newco');
      insert(a);

      Survey__c obj = new Survey__c(name='Client Satisfaction Survey');
      insert obj;

      User u = [Select Id, Email from User where Id =:context.installerID()];
      String toAddress= u.Email;
      String[] toAddresses = new String[]{toAddress};
      Messaging.SingleEmailMessage mail =
        new Messaging.SingleEmailMessage();
      mail.setToAddresses(toAddresses);
      mail.setReplyTo('support@package.dev');
      mail.setSenderDisplayName('My Package Support');
      mail.setSubject('Package install successful');
      mail.setPlainTextBody('Thanks for installing the package.');
      Messaging.sendEmail(new Messaging.Email[] { mail });
      }
    else
      if(context.previousVersion().compareTo(new Version(1,0)) == 0) {
      Survey__c obj = new Survey__c(name='Upgrading from Version 1.0');
      insert(obj);
      }
    if(context.isUpgrade()) {
      Survey__c obj = new Survey__c(name='Sample Survey during Upgrade');
      insert obj;
```

```
      }
    if(context.isPush()) {
      Survey__c obj = new Survey__c(name='Sample Survey during Push');
      insert obj;
      }
    }
  }
```

You can test a post install script using the new `testInstall` method of the `Test` class. This method takes the following arguments.

- A class that implements the `InstallHandler` interface.
- A `Version` object that specifies the version number of the existing package.
- An optional Boolean value that is `true` if the installation is a push. The default is `false`.

This sample shows how to test a post install script implemented in the `PostInstallClass` Apex class.

```
@isTest
static void testInstallScript() {
  PostInstallClass postinstall = new PostInstallClass();
    Test.testInstall(postinstall, null);
    Test.testInstall(postinstall, new Version(1,0), true);
    List<Account> a = [Select id, name from Account where name ='Newco'];
    System.assertEquals(a.size(), 1, 'Account not found');
  }
```

Specifying a Post Install Script

Once you have created and tested the post install script, you can specify it in the **Post Install Script** lookup field on the Package Detail page. In subsequent patch releases, you can change the contents of the script but not the Apex class.

The class selection is also available via the Metadata API as `Package.postInstallClass`. This is represented in package.xml as a `<postInstallClass>foo</postInstallClass>` element.

Running Apex on Package Uninstall

App developers can specify an Apex script to run automatically after a subscriber uninstalls a managed package. This makes it possible to perform cleanup and notification tasks based on details of the subscriber's organization. For simplicity, you can only specify one uninstall script. It must be an Apex class that is a member of the package.

The uninstall script is subject to default governor limits. It runs as a special system user that represents your package, so all operations performed by the script will appear to be done by your package. You can access this user by using UserInfo. You will only see this user at runtime, not while running tests.

If the script fails, the uninstall continues but none of the changes performed by the script are committed. Any errors in the script are emailed to the user specified in the **Notify on Apex Error** field of the package. If no user is specified, the uninstall details will be unavailable.

The uninstall script has the following restrictions. You can't use it to initiate batch, scheduled, and future jobs, to access Session IDs, or to perform callouts.

IN THIS SECTION:

Specifying an Uninstall Script

How does an Uninstall Script Work?

An uninstall script is an Apex class that implements the `UninstallHandler` interface. This interface has a single method called `onUninstall` that specifies the actions to be performed on uninstall.

```
global interface UninstallHandler {
  void onUninstall(UninstallContext context)
}
```

The `onUninstall` method takes a context object as its argument, which provides the following information.

- The org ID of the organization in which the uninstall takes place.
- The user ID of the user who initiated the uninstall.

The context argument is an object whose type is the `UninstallContext` interface. This interface is automatically implemented by the system. The following definition of the `UninstallContext` interface shows the methods you can call on the context argument.

```
global interface UninstallContext {
  ID organizationId();
  ID uninstallerId();
}
```

Example of an Uninstall Script

The sample uninstall script below performs the following actions on package uninstall.

- Inserts an entry in the feed describing which user did the uninstall and in which organization
- Creates and sends an email message confirming the uninstall to that user

```
global class UninstallClass implements UninstallHandler {
  global void onUninstall(UninstallContext ctx) {
    FeedItem feedPost = new FeedItem();
    feedPost.parentId = ctx.uninstallerID();
    feedPost.body = 'Thank you for using our application!';
    insert feedPost;

    User u = [Select Id, Email from User where Id =:ctx.uninstallerID()];
    String toAddress= u.Email;
    String[] toAddresses = new String[] {toAddress};
    Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
    mail.setToAddresses(toAddresses);
    mail.setReplyTo('support@package.dev');
    mail.setSenderDisplayName('My Package Support');
    mail.setSubject('Package uninstall successful');
    mail.setPlainTextBody('Thanks for uninstalling the package.');
    Messaging.sendEmail(new Messaging.Email[] { mail });
  }
}
```

You can test an uninstall script using the `testUninstall` method of the `Test` class. This method takes as its argument a class that implements the `UninstallHandler` interface.

This sample shows how to test an uninstall script implemented in the `UninstallClass` Apex class.

```
@isTest
static void testUninstallScript() {
  Id UninstallerId = UserInfo.getUserId();
  List<FeedItem> feedPostsBefore =
    [SELECT Id FROM FeedItem WHERE parentId=:UninstallerId AND CreatedDate=TODAY];
  Test.testUninstall(new UninstallClass());
  List<FeedItem> feedPostsAfter =
    [SELECT Id FROM FeedItem WHERE parentId=:UninstallerId AND CreatedDate=TODAY];
  System.assertEquals(feedPostsBefore.size() + 1, feedPostsAfter.size(),
    'Post to uninstaller failed.');
}
```

Specifying an Uninstall Script

Once you have created and tested the uninstall script and included it as a member of your package, you can specify it in the **Uninstall Script** lookup field on the Package Detail page. In subsequent patch releases, you can change the contents of the script but not the Apex class.

The class selection is also available via the Metadata API as `Package.uninstallClass`. This is represented in package.xml as an `<uninstallClass>foo</uninstallClass>` element.

Developing App Documentation

Salesforce recommends publishing your app on AppExchange with the following types of documentation:

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

Available in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure Option

You can include a **Configure** option for installers. This option can link to installation and configuration details, such as:

- Provisioning the external service of a composite app
- Custom app settings

The **Configure** option is included in your package as a custom link. You can create a custom link for your home page layouts and add it to your package.

1. Create a custom link to a URL that contains configuration information or a Visualforce page that implements configuration. When you create your custom link, set the display properties to `Open in separate popup window` so that the user returns to the same Salesforce page when done.
2. When you create the package, choose this custom link in the `Configure Custom Link` field of your package detail.

Data Sheet

Give installers the fundamental information they need to know about your app before they install.

Customization and Enhancement Guide

Let installers know what they must customize after installation as part of their implementation.

Custom Help
You can provide custom help for your custom object records and custom fields.

SEE ALSO:

Understanding Packages

Assigning Salesforce AppExchange Publishers

Assigning Salesforce AppExchange Publishers

Users that publish packages on the AppExchange must have the following user permissions:

Create Salesforce AppExchange packages
Allows a user to create packages and add components to it.

Upload Salesforce AppExchange Packages
Allows a user to upload and register or publish packages to the AppExchange.

The System Administrator profile automatically has both these permissions. Determine which of your users should have these permissions and add them to the appropriate user profiles or permission sets.

SEE ALSO:

Understanding Packages

Developing App Documentation

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

Available in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To assign permissions:
- Customize Application

Convert Unmanaged Packages to Managed

Your organization may already have uploaded and registered packages on Salesforce AppExchange directory. All packages uploaded prior to the Winter '07 release are unmanaged, meaning they cannot be upgraded in the installer's organization. You can convert them to managed packages by configuring your organization for managed packages and choosing a package to be managed. This allows you to notify installers when an upgrade is ready for them to install.

Before you convert an existing package to managed, notify your current installers of how to save their data:

1. Export all the data from the previous, unmanaged version of the package.
2. Uninstall the unmanaged package.
3. Install the new managed version of the package.
4. Import all the exported data into the new managed package.

Note: Note to installers: if you have made customizations to an installation of an unmanaged package, make a list of these customizations before uninstalling since you may want to implement them again. However, some customizations will not be possible with a managed package. See the Lightning Platform Quick Reference for Developing Packages.

To convert an unmanaged package into a managed package:

1. Enable managed packages for your organization.
2. From Setup, enter *Packages* in the Quick Find box, then select **Packages**.
3. Edit the package that you want to make managed, then select **Managed**.

Warning: Converting an unmanaged package to managed requires registering a namespace prefix that affects the API names of uniquely named package components such as custom fields or s-controls. S-controls stored in the s-control library or the Documents tab that do not use the Lightning Platform API still function properly after you register a namespace prefix. However, s-controls stored outside of your organization or s-controls that use the Lightning Platform API to call Salesforce may require some fine-tuning. For more information, see S-control in the *Object Reference*.

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To configure developer settings:
- Customize Application

To create packages:
- Create AppExchange Packages

To upload packages:
- Upload AppExchange Packages

SEE ALSO:

Manage Packages

Creating Managed Packages

Distribute Your Apps

Upload your apps, publish extensions and upgrades, and manage versions.

IN THIS SECTION:

Prepare Your Apps for Distribution

Why Use Trialforce?

Create Sign-Ups Using the API

Use API calls to the SignupRequest object to create sign-ups for prospective customers.

Publishing Extensions to Managed Packages

Publish Upgrades to Managed Packages

As a publisher, first ensure that your app is upgradeable by converting it to a managed package.

Manage Versions

Create and Upload Patches

Publishing Packages FAQ

Prepare Your Apps for Distribution

When you're ready to distribute your package, determine if you want to release a managed or unmanaged package. For more information about the different types of releases, see the *ISVforce Guide*.

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create packages:
- Create AppExchange Packages

To upload packages:
- Upload AppExchange Packages

1. Create a package:
   a. From Setup, enter *Packages* in the `Quick Find` box, then select **Packages**.
   b. Click **New**.
   c. Enter a name for your package. You can use a different name than what appears on AppExchange.
   d. From the dropdown menu, select the default language of all component labels in the package.
   e. Optionally, choose a custom link from the `Configure Custom Link` field to display configuration information to installers of your app. You can select a predefined custom link to a URL or s-control that you have created for your home page layouts; see the Configure Option on page 75. The custom link displays as a **Configure** link within Salesforce on the Salesforce AppExchange Downloads page and app detail page of the installer's organization.
   f. Optionally, in the `Notify on Apex Error` field, enter the username of the person to notify if an uncaught exception occurs in the Apex code. If you do not specify a username, all uncaught exceptions generate an email notification that is sent to Salesforce. This option is only available for managed packages. For more information, see *Handling Apex Exceptions in Managed Packages*.

      Note: Apex can only be packaged from Developer, Enterprise, Unlimited, and Performance Edition organizations.

   g. Optionally, in the `Notify on Packaging Error` field, enter the email address of the person who receives an email notification if an error occurs when a subscriber's attempt to install, upgrade, or uninstall a packaged app fails. This field appears only if packaging error notifications are enabled. To enable notifications, contact your Salesforce representative.
   h. Optionally, enter a description that describes the package. You can change this description before you upload it to AppExchange.
   i. Optionally, specify a post install script. You can run an Apex script in the subscriber organization after the package is installed or upgraded. For more information, see Running Apex on Package Install/Upgrade.
   j. Optionally, specify an uninstall script. You can run an Apex script in the subscriber organization after the package is uninstalled. For more information, see Running Apex on Package Uninstall.
   k. Click **Save**.

2. Salesforce sets your package API access privileges to `Unrestricted`. You can change this setting to further restrict API access of Salesforce components in the package. For more information, see *Manage API and Dynamic Apex Access in Packages on page 56*.
3. Add the necessary components for your app.
   a. Click **Add Components**.
   b. From the dropdown list, choose the type of component you want to add to your package.
      - At the top of the list, click a letter to display the contents of the sorted column that begin with that character.
      - If available, click the **Next Page** (or **Previous Page**) link to go to the next or previous set of components.
      - If available, click **fewer** or **more** at the bottom of the list to view a shorter or longer display list.
   c. Select the components you want to add.

      Note: Some components cannot be added to Managed - Released packages. For a list of these components, see *Developing Packages for Distribution*.

      S-controls cannot be added to packages with restricted API access.

   d. Click **Add To Package**.
   e. Repeat these steps until you have added all the components you want in your package.

   Note:
   - Some related components are automatically included in the package even if they don't display in the Package Components list. For example, when you add a custom object to a package, its custom fields, page layouts, and relationships with standard objects are automatically included. For a complete list of components Salesforce automatically includes, see the *ISVforce Guide*.
   - When you package a joined report, each block is included in the package. Although the blocks appear in the package as reports, when you click on a block, an error message indicates that you have "insufficient privileges" to view the report. This is expected behavior. Instead, click the name of the joined report to run it.

4. Optionally, click **View Dependencies** and review a list of components that rely on other components, permissions, or preferences within the package. An entity can include such things as an s-control, a standard or custom field, or an organization-wide setting like multicurrency. Your package cannot be installed unless the installer has the listed components enabled or installed. For more information on dependencies, see Understanding Dependencies on page 58. Click **Done** to return to the Package detail page.

   Note: You cannot upload packages that contain any of the following:
   - Workflow rules or workflow actions (such as field updates or outbound messages) that reference record types.
   - Reports that reference record types on standard objects.

5. Click **Upload**.

   Note: If you are creating a managed package to publish on AppExchange, you must certify your application before you package it. For more information, see Security Review on AppExchange.

6. On the Upload Package page, do the following:
   a. Enter a `Version Name`. As a best practice, it's useful to have a short description and the date.
   b. Enter a `Version Number` for the upload, such as *`1.0`*. The format is *`majorNumber.minorNumber`*.

      Note: If you're uploading a new patch version, you can't change the patch number.

The version number represents a release of a package. This field is required for managed and unmanaged packages. For a managed package, the version number corresponds to a Managed - Released upload. All beta uploads use the same version number until you upload a Managed - Released package version with a new version number. For example, the following is a sequence of version numbers for a series of uploads.

Upload Sequence	Type	Version Number	Notes
First upload	Managed - Beta	1.0	The first Managed - Beta upload.
Second upload	Managed - Released	1.0	A Managed - Released upload. The version number does not change.
Third upload	Managed - Released	1.1	Note the change of minor release number for this Managed - Released upload.
Fourth upload	Managed - Beta	2.0	The first Managed - Beta upload for version number 2.0. Note the major version number update.
Fifth upload	Managed - Released	2.0	A Managed - Released upload. The version number does not change.

c. For managed packages, select a `Release Type`:

- Choose Managed - Released to upload an upgradeable version. After upload, some attributes of Salesforce components are locked.
- Choose Managed - Beta if you want to upload a version of your package to a small sampling of your audience for testing purposes. You can still change the components and upload other beta versions.

  Note: Beta packages can only be installed in Developer Edition or sandbox organizations, and thus can't be pushed to customer organizations.

d. Change the `Description`, if necessary.

e. Optionally, specify a link to release notes for the package. Click **URL** and enter the details in the text field that appears. This link will be displayed during the installation process, and on the Package Details page after installation.

Note: As a best practice, point to an external URL, so you can make the information available to customers before the release, and update it independently of the package.

f. Optionally, specify a link to post install instructions for the package. Click **URL** or **Visualforce page** and enter the details in the text field that appears. This link will be displayed on the Package Details page after installation.

Note: As a best practice, point to an external URL, so you can update the information independently of the package.

g. Optionally, enter and confirm a password to share the package privately with anyone who has the password. Don't enter a password if you want to make the package available to anyone on AppExchange and share your package publicly.

h. Salesforce automatically selects the requirements it finds. In addition, select any other required components from the `Package Requirements` and `Object Requirements` sections to notify installers of any requirements for this package.

i. Click **Upload**.

7. After your upload is complete you can do any of the following.

- Click **Change Password** link to change the password option.
- Click **Deprecate** to prevent new installations of this package while allowing existing installations to continue operating.

  Note: You cannot deprecate the most recent version of a managed package.

  When you deprecate a package, remember to remove it from AppExchange as well. See "Removing Apps from AppExchange" in the AppExchange online help.

- Click **Undeprecate** to make a deprecated version available for installation again.

You receive an email that includes an installation link when your package has been uploaded successfully.

Note:

- When using the install URL, the old installer is displayed by default. You can customize the installation behavior by modifying the installation URL you provide your customers.
  - To access the new installer, append the text `&newui=1` to the installation URL.
  - To access the new installer with the "All Users" option selected by default, append the additional text `&p1=full` to the installation URL.
- If you uploaded from your Salesforce production org, notify installers who want to install it in a sandbox org to replace the "login.salesforce.com" portion of the installation URL with "test.salesforce.com."

SEE ALSO:

Understanding Packages
Manage Packages
Understanding Dependencies
Manage Versions
Create and Upload Patches
Publish Upgrades to Managed Packages
Publishing Extensions to Managed Packages

Why Use Trialforce?

Trialforce lets you provision a free trial of your offering quickly and easily. Each time a trial is provisioned, Trialforce creates a lead in the License Management App, which helps you track usage and convert prospects into paying customers. With Trialforce, you can:

- Run your own marketing campaign to maximize customer reach and adoption.
- Customize your offering, including branding, functionality, design, data, and trial experience.
- Manage trials for multiple offerings, versions, and editions from one convenient place.
- Let customers, including non-admin users, try your app or component without logging in to their production environment.

IN THIS SECTION:

Setting Up Custom Branding for Trialforce

App developers using Trialforce to create new trials of their product can optionally set up a branded login site and system emails. By branding these areas with your company's look and feel, users of your application are immersed in your brand from sign-up to login. Use custom branding for non-CRM apps, not for apps that extend Salesforce CRM and require Salesforce standard objects, such as Leads, Opportunities, and Cases.

Trialforce Source Organizations

New Trialforce Source Organization

Edit Trialforce Source Organization

Trialforce

Setting Up Custom Branding for Trialforce

App developers using Trialforce to create new trials of their product can optionally set up a branded login site and system emails. By branding these areas with your company's look and feel, users of your application are immersed in your brand from sign-up to login. Use custom branding for non-CRM apps, not for apps that extend Salesforce CRM and require Salesforce standard objects, such as Leads, Opportunities, and Cases.

A branded login page enables you to specify your login domain and login site.

- A login domain ends with `.cloudforce.com`, so if your company name is "mycompany," your login domain is `mycompany.cloudforce.com`.
- Your custom login site includes your text and company logo and mobile-friendly versions of your login site.

Branded emails allow you to specify fields in system-generated emails so that your company name, address, and other pertinent details are used in email correspondence. You can create multiple branded email sets for different campaigns or customer segments.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To manage Trialforce:
- Customize Application

Note: To configure branding, you must be logged in to a Trialforce Management Organization (TMO). To get your TMO, log a case in the Partner Community. Branding is not available for Trialforce Source Orgs created in the Environment Hub.

IN THIS SECTION:

Trialforce Branded Login Site

Trialforce Branded Email Sets

Trialforce Branded Login Site

Use the Trialforce Branded Login Site page to create, publish, and edit a login page that has your company's look and feel.

- If you haven't set up a login site yet, click **Set Up Login Site**.
- If you've already set up a login site, click **Publish** to make the site available, or **Launch Site Editor** to make changes.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To define package branding:
- Package Branding

IN THIS SECTION:

Trialforce Login Site Domain

Creating a Branded Login Page

Trialforce Login Branding Editor

Trialforce Login Site Domain

Choose a subdomain where customers will log into your application. Usually the subdomain is the name of your company.

1. In the field provided, enter a name.
2. Click **Check Availability**.
3. Accept the terms of use.
4. Click **Save and Launch Editor**.

Creating a Branded Login Page

Customers typically log in to your app using the traditional login.salesforce.com site. A branded login page enables you to customize this domain and parts of this login page so you can provide a branded experience for your customers. Your custom login site includes your text and company logo, and mobile-friendly versions of your login site as well.

To create a branded login page:

1. Log in to your Trialforce Management Organization.
2. From Setup, enter *Login Site* in the Quick Find box, then select **Login Site**.
3. Click **Set Up Login Site**.
4. Select a subdomain for your login site by providing a name in the field provided. Usually this is the name of your company.

   Note: A login domain ends with .cloudforce.com, so if your company name is "mycompany," your login domain is mycompany.cloudforce.com.

5. Check the availability of the domain and then accept the terms of use.
6. Click **Save and Launch Editor**.
7. Use the Login Brand Editor to change how your login page looks. For additional help using the editor, click **Help for this Page**.
8. Click **Save and Close**.
9. If you're ready to make these changes available to your TSO, click **Publish**. Otherwise your changes are saved and you can publish later.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To manage Trialforce:
- Customize Application

Trialforce Login Branding Editor

Use the Login Branding Editor to design your login pages.

1. Log into your Trialforce Management Organization.
2. From Setup, enter *Login Site* in the Quick Find box, then select **Login Site**.
3. At the top of the editor, click the tab for the login page size: **Desktop** or **Mobile**.
4. In the left pane, expand the Page Header node and click **Select File** to choose your company logo for each size screen your app supports.
5. In the **Logo Link** > **Use custom link** field, optionally, enter a web address to be used when a customer clicks your logo, such as your corporate website. The URL must start with http:// or https://. If you leave this field blank, your logo will not have a link.
6. Expand the Page Content node and paste the URL of the trial sign-up link into the **Trial Sign-Up Link** > **Use custom link** field. This is the link your prospects will click to request a free trial on your website. Typically, ISVs create a separate sign-up page for this purpose.
7. Provide URLs for the right and bottom of the page. If you leave these fields blank, the frames default to the ones used on the Salesforce login page.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To manage Trialforce:
- Customize Application

8. Expand the page footer and provide your company name and font color.
9. Expand the page background node and provide a background image and color.
10. At the top of the page click **Save and Close**.
11. On the Branded Login Site page, click **Preview** for the size of the page you want to see. Make sure your login page appears correct for each login page your app supports.

Trialforce Branded Email Sets

Trialforce Email Branding allows you to modify system-generated emails so that they appear to come from your company rather than from Salesforce. Trialforce Email Branding only applies to users who sign up for your application through Trialforce.

Each Trialforce source organization comes with a standard set of email notifications that are sent to customers. For example, customers get an email notification when they first sign up, or when they reset their password. You don't have to rewrite all of these system-generated emails yourself. Just provide the values for the fields and the system takes care of the rest.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To define package branding:
- Package Branding

IN THIS SECTION:

Edit Trialforce Branded Email Set

Edit Trialforce Branded Email Set

To begin, click **New Email Set** or **Edit** next to an existing email set.

1. Fill in the fields with your company info.
2. In the Preview Emails area, click through the different types of generated emails and make sure they read correctly.
3. Click **Save**.
4. If you're ready to make these emails available to your Trialforce source organizations, click **Publish**. Otherwise your changes are saved and you can publish later.

Trialforce Source Organizations

The Trialforce Source Organizations page helps you create and manage your Trialforce source organizations.

- To create a new source organization, click **New** .
- If you have an existing source organization you want to use, click **Login**.
- To edit an existing source organization, click **Edit**.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

New Trialforce Source Organization

To create a new source organization:

1. Enter a new username and email address for the administrator account.
2. Enter the source organization name and select the branding.
3. Click **Create**.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To define package branding:
- Package Branding

Edit Trialforce Source Organization

To edit a Trialforce source organization:

1. Enter the source organization name and select the branding.
2. Click **Save**.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To define package branding:
- Package Branding

Trialforce

To create a Trialforce template:

1. Click **New Trialforce Template**.
2. Specify a description of the template and whether to include data in the dialog that appears.
3. Click **Save**.

You receive an email with the ID of the new template after it's generated. Submit the template for review before you can use it to sign up trial organizations. Remember to generate a new template each time you make updates to your TSO so that your trials always reflect the most recent state.

Note: You can create a template only if your TSO is less than or equal to 1 GB.

Each template has a status with one of the following values.

In Progress
When a template is first created, it always has this status. It then moves to either Success or Error status.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To define package branding:
- Package Branding

Success
The template can be used to create trial organizations.

Error
The template cannot be used because something has gone wrong and debugging is required.

Deleted
The template is no longer available for use. Deleted templates are removed during system updates.

Create Sign-Ups Using the API

Use API calls to the SignupRequest object to create sign-ups for prospective customers.

Using API calls to the SignupRequest object, you can collect and analyze detailed information on all sign-ups from your business organization. You have control over the sign-up process and enhanced visibility into your prospective customers. For example, you can:

- Run reports and collect metrics, such as the number of sign-ups per day or the number of sign-ups in different countries.
- Customize the SignupRequest object to add fields of special interest to your company.
- Create triggers to initiate specific actions, such as send an email notification when a new sign-up request is made.
- Enable sign-ups from a wide range of client applications and devices, so you have more channels for customer acquisition.

For more information on working with objects, see the Object Reference for Salesforce and Lightning Platform.

USER PERMISSIONS

To create or view sign-up requests:
- SignupRequest API

IN THIS SECTION:

Signup Request Home

Note: You are limited to 20 signups per day. To make additional signups, log a case in the Partner Community.

The Signup Requests tab displays the signup requests home page. From this page, you can perform the following actions.

- Create a new signup. If you using a Trialforce template to create the signup, make sure the template has been approved.
- View the details of a previous signup, including its history and approval status.
- Create new views to display signups matching criteria that you specify.

USER PERMISSIONS

To create or view signup requests:
- Signup Request API

Create a Signup Request

1. Select **Signup Request** from the Create New drop-down list in the sidebar, or click **New** next to **Recent Signup Requests** on the signup requests home page.
2. Enter the information for the signup request.

USER PERMISSIONS

To create or view signup requests:
- Signup Request API

3. Click **Save** when you're finished, or click **Save & New** to save the current signup request and add another.

Viewing Signup Request Details

From the Signup Request detail page:

- Click **Delete** to delete the signup request
- Click **Clone** to create a new signup request with the same attributes as this one

The detail page has the following sections.

- Signup Request Detail
- Signup Request History

USER PERMISSIONS

To create or view signup requests:

- Signup Request API

Signup Request Detail

This section displays the following attributes (in alphabetical order).

Attribute	Description
`Company`	The name of the company requesting the trial signup.
`Country`	The two-character, upper-case ISO-3166 country code. You can find a full list of these codes at a number of sites, such as: www.iso.ch/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html
`Created Org`	The 15–character Organization ID of the trial organization created. This is a read-only field provided by the system once the signup request has been processed.
`Email`	The email address of the admin user for the trial signup.
`Error Code`	The error code if the signup request isn't successful. This is a read-only field provided by the system to be used for support purposes.
`First Name`	The first name of the admin user for the trial signup.
`Last Name`	The last name of the admin user for the trial signup.
`Edition`	The Salesforce template that is used to create the trial organization. Possible values are `Partner Group`, `Professional`, `Partner Professional`, `Sales Enterprise`, `Professional TSO`, `Enterprise`, `Partner Enterprise`, `Service Professional`, `Enterprise TSO`, `Developer`, and `Partner Developer`.
`Preferred Language`	The language of the trial organization being created. Specify the language using a language code listed under Fully Supported Languages in Supported Languagesin Salesforce Help. For example, use *`zh_CN`* for simplified Chinese. The value you select overrides the language set by the locale. If you specify an invalid language, the organization defaults to the default language of the country.
	Populated during the sign-up request and for internal use by Salesforce.
`ShouldConnectToEnvHub`	When set to `true`, the trial organization is connected to the Environment Hub. The sign-up must take place in the hub master organization or a spoke organization.
`Source Org`	The 15–character Organization ID of the Trialforce Source Organization from which the Trialforce template was created.

Attribute	Description
`Status`	The status of the request. Possible values are `New`, `In Progress`, `Error`, or `Success`. The default value is `New`.
`Template`	The 15–character ID of the approved Trialforce template that is the basis for the trial signup. The template is required and must be approved by Salesforce.
`Template Description`	The description of the approved Trialforce template that is the basis for the trial signup.
`Trial Days`	The duration of the trial signup in days. Must be equal to or less than the trial days for the approved Trialforce template. If not provided, it defaults to the trial duration specified for the Trialforce template.
`Username`	The username of the admin user for the trial signup. It must follow the address convention specified in RFC822: www.w3.org/Protocols/rfc822/#z10

Signup Request History

This section shows the date the signup request was created, the user who created it, and the actions that have been performed on it.

Publishing Extensions to Managed Packages

An *extension* is any package, component, or set of components that adds to the functionality of a managed package. An extension requires that the base managed package be installed in the organization. For example, if you have built a recruiting app, an extension to this app might include a component for performing background checks on candidates.

The community of developers, users, and visionaries building and publishing apps on Salesforce AppExchange is part of what makes Lightning Platform such a rich development platform. Use this community to build extensions to other apps and encourage them to build extensions to your apps.

To publish extensions to a managed package:

1. Install the base package in the Salesforce organization that you plan to use to upload the extension.
2. Build your extension components.

   Note: To build an extension, install the base package and include a dependency to that base package in your package. The extension attribute will automatically become active.
3. Create a new package and add your extension components. Salesforce automatically includes some related components.
4. Upload the new package that contains the extension components.
5. Proceed with the publishing process as usual. For information on creating a test drive or registering and publishing your app, go to http://sites.force.com/appexchange/publisherHome.

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

Available in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create packages:
- Create AppExchange Packages

To upload packages:
- Upload AppExchange Packages

Note: Packages cannot be upgraded to Managed - Beta if they are used within the same organization as an extension.

SEE ALSO:

- Prepare Your Apps for Distribution
- Understanding Dependencies
- Manage Versions
- Publish Upgrades to Managed Packages

Publish Upgrades to Managed Packages

As a publisher, first ensure that your app is upgradeable by converting it to a managed package.

Any changes you make to the components in a managed package are automatically included in subsequent uploads of that package, with one exception. When you upgrade a package, changes to the API access are ignored even if the developer specified them. This ensures that the administrator installing the upgrade has full control. Installers should carefully examine the changes in package access in each upgrade during installation and note all acceptable changes. Then, because those changes are ignored, the admintrator should manually apply any acceptable changes after installing an upgrade.

1. From Setup, enter *Packages* in the Quick Find box, then select **Packages**.
2. Select the package from the list of available packages.
3. View the list of package components. Changes you have made to components in this package are automatically included in this list. If the changes reference additional components, those components are automatically included as well. To add new components, click **Add** to add them to the package manually.
4. Click **Upload** and upload it as usual.

   After you upload a new version of your Managed - Released package, you can click **Deprecate** so installers cannot install an older version. Deprecation prevents new installations of older versions without affecting existing installations.

   You cannot deprecate the most recent version of a managed package upload.
5. When you receive an email with the link to the upload on Salesforce AppExchange, notify your installed users that the new version is ready. Use the list of installed users from the License Management Application (LMA) to distribute this information. The License Management Application (LMA) automatically stores the version number that your installers have in their organizations.

SEE ALSO:

- Prepare Your Apps for Distribution
- Understanding Dependencies
- Manage Versions
- Create and Upload Patches
- Publishing Extensions to Managed Packages

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Developer** Edition

Package uploads and installs are available in **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To configure developer settings:

- Customize Application

To create packages:

- Create AppExchange Packages

To upload packages:

- Upload AppExchange Packages

Manage Versions

After you upload a package to the AppExchange, you can still manage it from Salesforce. To manage your versions:

1. From Setup, enter *Packages* in the Quick Find box, then select **Packages**.
2. Select the package that contains the app or components you uploaded.
3. Select the version number listed in the Versions tab.
   - Click **Change Password** link to change the password option.
   - Click **Deprecate** to prevent new installations of this package while allowing existing installations to continue operating.

     Note: You cannot deprecate the most recent version of a managed package.

     When you deprecate a package, remember to remove it from AppExchange as well. See "Removing Apps from AppExchange" in the AppExchange online help.
   - Click **Undeprecate** to make a deprecated version available for installation again.

Note: To create a test drive or choose a License Management Organization (LMO) for what you have uploaded, click **Proceed to AppExchange** from the package upload detail page.

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

Available in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To upload packages:
- Upload AppExchange Packages

SEE ALSO:

Prepare Your Apps for Distribution
Understanding Dependencies
Create and Upload Patches
Publish Upgrades to Managed Packages
Publishing Extensions to Managed Packages
Security Guidelines for Apex and Visualforce Development

Create and Upload Patches

Patch versions are developed and maintained in a patch development org. .

To create a patch version:

1. From Setup, enter *Packages* in the Quick Find box, then select **Packages**.
2. Click the name of your managed package.
3. On the Patch Organization tab, click **New**.
4. Select the package version that you want to create a patch for in the Patching Major Release dropdown. The release type must be Managed - Released.
5. Enter a username for a login to your patch org.
6. Enter an email address associated with your login.
7. Click **Save**.

EDITIONS

Available in: **Developer** Edition

USER PERMISSIONS

To push an upgrade or create a patch development org
- Upload AppExchange Packages

Note: If you ever lose your login information, click **Reset** on the package detail page under Patch Development Organizations to reset the login to your patch development org.

If the main development org from which you created the patch org has My Domain enabled, the patch org also has My Domain enabled. The name of the patch development org's custom subdomain is randomly generated.

After you receive an email that Salesforce has created your patch development org, you can click **Login** to begin developing your patch version.

Development in a patch development org is restricted.

- You can't add package components.
- You can't delete existing package components.
- API and dynamic Apex access controls can't change for the package.
- No deprecation of any Apex code.
- You can't add new Apex class relationships, such as `extends`.
- You can't add Apex access modifiers, such as `virtual` or `global`.
- You can't add new web services.
- You can't add feature dependencies.

When you finish developing your patch, upload it through the UI in your patch development org. You can also upload a package using the Tooling API. For sample code and more details, see the PackageUploadRequest object in the *Tooling API Developer Guide*.

Note: When you upload a new package in your patch development org, the upload process is asynchronous. Because the time to process the request varies, the package might not be available immediately after the upload. While waiting, you can run SOQL queries on the PackageUploadRequest status field to monitor the request.

1. From Setup, enter *`Packages`* in the Quick Find box, then select **Packages**.
2. Click the name of the package.
3. On the Upload Package page, click **Upload**.
4. Enter a `Version Name`. As a best practice, it's useful to have a short description and the date.
5. Notice that the `Version Number` has had its `patchNumber` incremented.
6. For managed packages, select a `Release Type`:
   - Choose Managed - Released to upload an upgradeable version. After upload, some attributes of Salesforce components are locked.
   - Choose Managed - Beta if you want to upload a version of your package to a small sampling of your audience for testing purposes. You can still change the components and upload other beta versions.

     Note: Beta packages can only be installed in Developer Edition or sandbox organizations, and thus can't be pushed to customer organizations.
7. Change the `Description`, if necessary.
8. Optionally, enter and confirm a password to share the package privately with anyone who has the password. Don't enter a password if you want to make the package available to anyone on AppExchange and share your package publicly.
9. Salesforce automatically selects the requirements it finds. In addition, select any other required components from the `Package Requirements` and `Object Requirements` sections to notify installers of any requirements for this package.
10. Click **Upload**.

Publishing Packages FAQ

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

IN THIS SECTION:

SEE ALSO:

Installing Packages FAQ

How are packages ranked?

AppExchange lists feedback and ratings similar to those found on most consumer Internet sites. This gives the power to the Salesforce user community to determine how useful a package is.

What types of things can I publish?

You can publish any collection of components, such as tabs, reports, and dashboards that together address a specific business need. You can bundle these components into a package to publish them together. For a list of components you can include in a package, see the Lightning Platform Quick Reference for Developing Packages. Regardless of what components you add to your package, your data is never included.

Support Subscribers of Your App

Support AppExchange customers, view subscriber organizations and details.

IN THIS SECTION:

Support Your AppExchange Customers

After you publish a solution on AppExchange, you're responsible for the end user support. Even when you've built the best solution the world has ever seen, customers need your help from time to time. Learn about the tools that you can use to attend to your customers' needs, grow your business, and affirm your reputation on AppExchange.

Note: When customers contact Salesforce Customer Support with questions about your solution, we direct them to your AppExchange listing. Make sure the contact information on your listing is accurate and complete.

Subscriber Organizations

This page shows a list of subscriber organizations with your package installed. To find a subscriber organization quickly, enter a subscriber name or organization ID in the search box and click **Search**. Click the name of a subscriber organization to view detailed information about it.

Note: Only subscribers who have installed at least one managed package that is linked to your LMA will appear in this list.

USER PERMISSIONS

To log in to subscriber organizations:
- Log in to Subscriber Organization

Viewing Subscriber Details

The Subscriber Overview page, accessed by clicking the organization's name from the **Subscribers** tab of the LMA, provides detailed information about each subscriber organization. This can give you insight into how a customer is using your app and help you in troubleshooting problems.

USER PERMISSIONS

To log in to subscriber organizations:
- Log in to Subscriber Organization

Under Organization Details:

- The name and contact information is in Setup, on the Company Information page in the subscriber's organization. This may differ from the information shown in your LMA lead, account, or contact records.
- Organization ID is a unique ID that identifies this customer's Salesforce organization.
- Instance determines which Salesforce data center this customer's organization resides in. It also determines when the customer will get upgraded with a new version of Salesforce. See trust.salesforce.com during the release period to understand which version of Salesforce the customer is using.

The page also includes these related lists.

Limits
Information on the file space, data space, and number of API requests associated with this customer, as a percentage.

Login Access Granted
A list of users who have granted login access and the date when access will expire.

Packages and Licensing
A list of all packages installed in this organization and associated with this LMA. For each package, it shows the version of the app a customer is currently using, the total number of licenses provisioned to the subscriber and the number they've used. This information should match the license record for the subscriber in your LMA.

Request Login Access from a Customer

Before logging in to a subscriber org, first request login access from the customer.

To request login access, ask the user to go to personal settings and click either **Grant Account Login Access** or **Grant Login Access**. If the publisher isn't listed, one of the following applies.

- A system admin disabled the ability for non-admins to grant access.
- The user doesn't have a license for the package.
- The package is licensed to the entire org. Only admins with the "Manage Users" permission can grant access.
- The org preference **Administrators Can Log in as Any User** is enabled.

Note: Unless the org preference **Administrators Can Log in as Any User** is enabled, access is granted for a limited amount of time, and the subscriber can revoke access at any time. Any changes you make while logged in as a subscriber are logged in the audit trail.

Log In to Subscriber Orgs

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To log in to subscriber orgs:
- Log in to Subscriber Org

Note: This feature is available only in orgs with a Salesforce Platform or full Salesforce license. You can't log in to subscriber orgs on Government Cloud instances.

Multi-Factor Authentication Requirement for Logins to Subscriber Orgs

In Summer '21 the Require Multi-Factor Authentication For Logins to Subscriber Orgs release update is enforced on all License Management Orgs (LMO). This update provides subscribers an extra layer of security by verifying the identity of the user accessing their org. It also provides you more control and accountability over which users log into a subscriber org.

To test and apply this update, from Setup, in the Quick Find box, enter `Release Updates`, and select **Release Updates**. Locate Require Multi-Factor Authentication For Logins to Subscriber Orgs and follow the testing and activation steps.

Log In to a Subscriber Org

After you've logged in to the LMO using Multi-Factor Authentication (MFA), and your subscriber has granted you login access, you're ready to log in.

1. In the License Management App (LMA), click the **Subscribers** tab.
2. To find a subscriber org, enter a subscriber name or org ID in the search box and click **Search**.
3. Click the name of the subscriber org.
4. On the Org Details page, click **Login** next to a user's name. You have the same permissions as the user you logged in as.
5. When you're finished troubleshooting, log out of the subscriber org.

Note: Some subscribers also require MFA in addition to the MFA required for the LMO. Ask your subscriber if their org requires MFA to log in.

In this second MFA scenario, your login attempt sends an MFA notification to your subscriber, and your login is blocked until your subscriber responds to the notification. To ensure your subscriber is available to respond to the MFA notification, consider coordinating a specific login time.

Best Practices

- Create an audit trial that indicates when and why a subscriber org login has occurred. You can create an audit trail by logging a case in your LMO before each subscriber org login.

- When you access a subscriber org, you're logged out of your LMO (License Management Organization). You can set up a my domain so that you aren't automatically logged out of your LMO when you log in to a subscriber org. To set up a My Domain subdomain, from Setup, enter *My Domain* in the Quick Find box, then select **My Domain**.
- Be careful to allow only trusted support and engineering personnel to log in to a subscriber's org. Since this feature can include full read/write access to customer data and configurations, it's vital to your reputation to preserve their security.
- Control who has access by giving the Log in to Subscriber Org user permission to specific support personnel, via a profile or permission set.

Troubleshooting in Subscriber Organizations

When logged in as a user in a subscriber's org, you have access that the subscriber doesn't have. You can view the obfuscated code in your Managed - Released packages, view logs that the subscriber can't see, and initiate ISV Customer Debugger sessions.

Troubleshoot with Debug Logs

The simplest way to debug your code in a subscriber's org is to generate Apex debug logs that contain the output from your managed packages. These logs include log lines that would normally not be exposed to the subscriber. Using this log information, you can troubleshoot issues that are specific to that subscriber.

1. If the user has access, set up a debug log: From Setup, enter *Debug Logs* in the Quick Find box, then select **Debug Logs**.
2. Launch the Developer Console.
3. Perform the operation and view the debug log with your output.

Subscribers are unable to see the logs you set up or generate since they contain your unobfuscated Apex code.

In addition, you can view and edit data contained in protected custom settings from your managed packages when logged in as a user.

Troubleshoot with the ISV Debugger

Each License Management Org can use one free ISV Debugger session at a time. The ISV Debugger is part of the Salesforce Extensions for Visual Studio Code. The ISV Debugger can be used only in sandbox orgs, so you can initiate debugging sessions only from a customer's sandbox.

For details, see the ISV Debugger documentation.

SEE ALSO:

Open the Developer Console