salesforce.com SOFTWARE
Success On Demand.™

Quick Reference for Developing Packages

Last updated: July 6, 2009

Table of Contents

Overview of Packages

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

This guide provides quick reference information that is useful when you are starting to design and develop packages. It explains basic packaging concepts as well as which components, such as custom objects, can be included in packages. See the *AppExchange Publishing Guide* for information on how to upload your packages after development is complete and more tips and best practices for preparing, building, and packaging your applications.

Packages come in two forms—managed and unmanaged:

- *Unmanaged packages* can be used for a one time distribution or migration of components to one or more Salesforce organizations. Once the components are distributed in an unmanaged package, the developer loses all control over the components and the installers can alter them as they see fit. Unmanaged packages are not upgradeable and are typically free. They are useful for distributing application templates to provide developers with the basic building blocks for an application.
- A collection of application components that are posted as a unit on AppExchange, and are associated with a namespace and a License Management Organization. A package must be managed for it to be published publicly on AppExchange, and for it to support upgrades. An organization can create a single managed package that can be downloaded and installed by many different organizations. They differ from unmanaged packages in that some components are locked, 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, so as to protect the intellectual property of the developer. Managed packages are ideal when you are building an app that you have plans to upgrade and sell. Managed packages also offer license management and intellectual property protection.

You need a Developer Edition organization to create a managed package. You can create an unmanaged package in Group, Professional, Enterprise, Unlimited, and Developer Editions. If you are not already a member of the developer community, go to `developer.force.com/join` and follow the instructions for signing up for a Developer Edition account.

A package version is a set of components uploaded in a package. It is denoted by a version number, *`majorNumber.minorNumber`*, such as 2.1.

What Is In a Package?

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.

Packages

A package is comprised of many components, such as custom objects, Apex classes and triggers, or apps. They can be managed or unmanaged. For a list of all the components that can be included in a package, see Packageable Components on page 4.

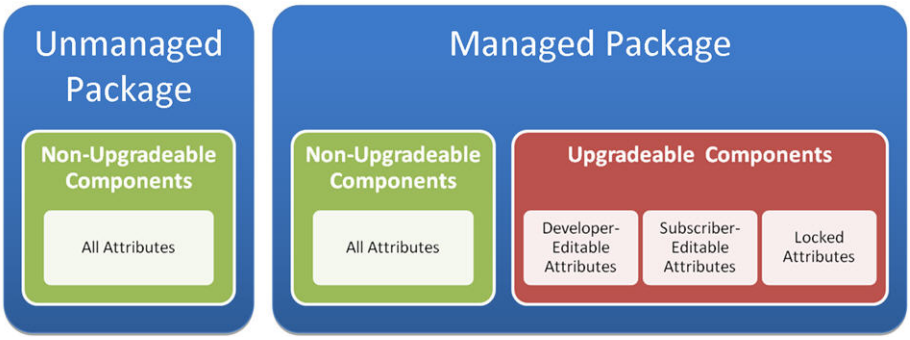
Components

A *component* is one constituent part of a package. A component 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 cannot.

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 and the subscriber. 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 cannot be edited by either the developer or subscriber. For information on which attributes are editable for each component, see Component Attributes in Managed Packages on page 9.

The following image illustrates these concepts.



Managed Package Statuses

When uploading a managed package, you must determine whether you are ready to release the contents of the package. Releasing a package has irreversible implications that you should understand prior to uploading a package. The following are the two types of uploads available for managed packages:

Managed - Beta

The purpose of a Managed - Beta package is to allow the developer to test their application in different Salesforce organizations and to share the app with a pilot set of users for evaluation and feedback. This type of package provides the developer with the most flexibility as any changes can be made to the package. However, subscribers to a beta package cannot upgrade their packages. The best practice is for subscribers to uninstall the beta package when testing is complete, then install the Managed - Released package.

A beta package cannot be installed into a company's production Salesforce organization. Users can install beta packages in sandbox, Developer Edition, or testing organizations only. Partners registered in the Force.com ISV Program receive access to free Enterprise and Professional Edition testing organizations that can be used for beta package testing. For free registration, see developer.force.com/isv.

Managed - Released

A package should only be uploaded as Managed - Released once all testing is complete and the package is ready for installation by customers. The released version of the package can be listed on the AppExchange and made publicly available. Once a Managed - Released version of a package is uploaded, the components in the package are also Managed - Released and certain destructive changes cannot be made by the developer. These restrictions ensure that subscribers can easily upgrade in the future.

Packageable Components

The Salesforce components in a managed package behave differently than those in an unmanaged package at different points in the release process. For example, a subscriber cannot change certain attributes of managed components in any package that is installed, regardless of whether it is beta. Likewise, the developer cannot change certain attributes of managed components previously uploaded in a Managed - Released state. However, Managed - Beta components have the same properties as unmanaged components in the developer's organization. Developers should determine when the right time is to make a beta package Managed - Released because doing so restricts the properties of each component within it. To understand each state of a managed package, see "Planning the Release of Managed Packages" in the Salesforce online help.

A package is comprised of many different types of components. A component may behave differently depending on the type of package. The table that follows shows the behavior of components for each of the properties listed below, such as Packaged Explicitly or Implicitly. Each property represents a column in the table. For information on the component attributes, see Component Attributes in Managed Packages on page 9.

Packaged Explicitly or Implicitly (Applies to managed and unmanaged packages)

Components can be added either explicitly or implicitly. For example, when you create a custom field on a standard object, you must add the custom field to your package explicitly. However, if you have created a custom object and added a custom field on it, when you add the custom object to the package the custom field is implicitly added to the package.

- **Explicitly**: The component can be directly included in the package.
- **Implicitly**: The component cannot be added directly to the package. The component is added to the package when another dependent component, usually a custom object, is added to the package.

Automatic Renaming (Applies to unmanaged packages)

Salesforce can resolve naming conflicts automatically on install.

- **No**: If a naming conflict occurs the install is blocked.
- **Yes**: If a naming conflict occurs Salesforce can optionally change the name of the component being installed.

Upgradeable (Applies to managed packages)

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

- **No**: The component is not upgraded.
- **Yes**: The component is upgraded.

Developer Deleteable (Applies to managed packages)

A developer can delete some components after the package is uploaded as Managed - Released. Deleted components are not deleted in the subscriber's organization during a package upgrade. For additional information on deleting components, see Protectable on page 5.

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

Subscriber Deleteable (Applies to managed packages)

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

- **No**: The subscriber cannot delete the component.

- **Yes**: The subscriber can delete the component.

Standard and Extension Objects Supported (Applies to managed and unmanaged packages)

Some components related to a standard object or a custom object installed by another managed package can be packaged, even though the object cannot be packaged. For example, a custom field on a standard object can be individually packaged, while the only way to include a custom field on a custom object created in the developer organization is to add the object to the package. An extension is any package that adds to the functionality of another managed package.

- **No**: The component cannot be packaged on standard and extension objects.
- **Yes**: The component can be packaged on standard and extension objects.
- **Not applicable**: The component has no relationship to a standard or extension object.

Protectable (Applies to managed packages)

Developers can mark certain components as protected. Protected components cannot be linked to or referenced by components created in a subscriber organization. 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 cannot delete it. When the subscriber upgrades to a version of the package where the component is deleted, the component is removed from the subscriber's organization.

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

IP Protection (Applies to managed packages)

Certain components automatically include intellectual property protection, such as obfuscating Apex code. The only exceptions are Apex methods declared as global, meaning that the method signatures can be viewed by the subscriber. The information in the components you package and publish may be visible to users on Force.com AppExchange. Use caution when adding your code to a custom s-control, formula, Visualforce page, or any other component that you cannot hide in your app.

- **No**: The component does not support intellectual property protection.
- **Yes**: The component supports intellectual property protection.

Component	Packaged Explicitly or Implicitly	Automatic Renaming	Upgradeable	Developer Deleteable	Subscriber Deleteable	Standard and Extension Objects Supported	Protectable	IP Protection
Analytic Snapshot	Explicitly	Yes	No	Yes	Yes	Yes	No	No
Apex Class	Explicitly	No	Yes	No	No	N/A	No	Yes
Apex Sharing Reason	Implicitly	No	Yes	No	No	No	No	No

Component	Packaged Explicitly or Implicitly	Automatic Renaming	Upgradeable	Developer Deleteable	Subscriber Deleteable	Standard and Extension Objects Supported	Protectable	IP Protection
Apex Sharing Recalculation	Implicitly	No	No	Yes	Yes	No	No	No
Apex Trigger	On a standard or extension object: Explicitly On an object in the package: Implicitly	No	Yes	No	No	Yes	No	Yes
Application	Explicitly	No	No	Yes	Yes	N/A	No	No
Custom Button or Link	On a standard object: Explicitly On a custom object: Implicitly	No	Yes	No	No	Yes	No, except custom links (for Home page only)	No
Custom Field	On a standard object: Explicitly On a custom object: Implicitly	No	Yes	No	No	Yes	No	No
Custom Label	Explicitly	No	Yes	Yes, if protected	No	N/A	Yes	No
Custom Object	Explicitly	No	Yes	No	No	N/A	No	No
Custom Report Type	Explicitly	No	Yes	No	No	Yes	No	No

Component	Packaged Explicitly or Implicitly	Automatic Renaming	Upgradeable	Developer Deleteable	Subscriber Deleteable	Standard and Extension Objects Supported	Protectable	IP Protection
Dashboard	Explicitly In a folder: Implicitly	Yes	No	Yes	Yes	N/A	No	No
Document	Explicitly In a folder: Implicitly	Yes	No	Yes	Yes	N/A	No	No
Email Template	Explicitly In a folder: Implicitly	Yes	No	Yes	Yes	N/A	No	No
Folder	Explicitly	Yes	No	Yes	Yes	N/A	No	No
Home Page Component	Explicitly	No	Yes	No	No	N/A	No	No
Home Page Layout	Explicitly	No	No	Yes	Yes	N/A	No	No
Letterhead	Explicitly	Yes	No	Yes	Yes	N/A	No	No
List View	Implicitly	Yes	No	Yes	Yes	No	No	No
Page Layout	Implicitly	No	No	Yes	Yes	No	No	No
Record Type	Implicitly	No	Yes	No	No	No	No	No
Report	Explicitly In a folder: Implicitly	Yes	No	Yes	Yes	N/A	No	No
S-Control	Explicitly	No	Yes	No	No	N/A	No	No
Static Resource	Explicitly	No	Yes	No	No	N/A	No	No
Tab	Explicitly	No	Yes	No	No	Yes	No	No
Translation	Explicitly	No	Yes	No	No	N/A	No	No
Validation Rule	Implicitly	No	Yes	No	No	No	No	No
Visualforce Component	Explicitly	No	Yes	No	No	Yes	No	Yes

Component	Packaged Explicitly or Implicitly	Automatic Renaming	Upgradeable	Developer Deleteable	Subscriber Deleteable	Standard and Extension Objects Supported	Protectable	IP Protection
Visualforce Page	Explicitly	No	Yes	No	No	Yes	No	No
Workflow Email Alert	Explicitly	No	Yes	Yes, if protected	No	Yes	Yes	No
Workflow Field Update	Explicitly	No	Yes	Yes, if protected	No	Yes	Yes	No
Workflow Outbound Message	Explicitly	No	Yes	Yes, if protected	No	Yes	Yes	No
Workflow Rule	Explicitly	No	Yes	No	No	Yes	No	No
Workflow Task	Explicitly	No	Yes	Yes, if protected	No	Yes	Yes	No

Component Attributes in Managed Packages

A component is composed of a series of *attributes*, or fields, that specify properties of the component, such as the `Description` attribute of a dashboard. The following table shows the behavior of attributes on components in a managed package. In an unmanaged package, all attributes can be edited by both the subscriber and the developer since the package cannot be upgraded. For information on the properties of packageable components, see Packageable Components on page 4.

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, they are not upgradeable; only new subscribers receive the latest changes.

Locked

Once a package is Managed - Released, the developer and subscriber cannot edit the component attributes in this column.

Component	Developer Editable	Subscriber and Developer Editable	Locked
Analytic Snapshot		• All attributes except Analytic Snapshot Unique Name	• Analytic Snapshot Unique Name
Apex Class	• API Version • Code		• Name
Apex Sharing Reason	• Reason Label		• Reason Name
Apex Sharing Recalculation		• Apex Class	
Apex Trigger	• API Version • Code		• Name
Application		• All attributes except App Name	• App Name
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

Component	Developer Editable	Subscriber and Developer Editable	Locked
Custom Field	• Auto-Number Display Format • Description • Default Value • Field Label • Formula • Related List Label • Roll-Up Summary Filter Criteria	• Help Text • Mask Type • Mask Character • Sharing Setting • Sort Picklist Values	• Child Relationship Name • Data Type • Decimal Places • External ID • Field Name • Length • Required • 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 • Development Status • Enable Divisions • Grant Access Using Hierarchy • Search Layouts • Track Field History	• Object Name • Record Name Data Type • Record Name Display Format
Custom Report Type	• All attributes except Development Status and Report Type Name	• Development Status	• Report Type Name
Dashboard		• All attributes except Dashboard Unique Name	• Dashboard Unique Name
Document		• All attributes except Document Unique Name	• Document Unique Name
Email Template		• All attributes except Email Template Name	• Email Template Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
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
Letterhead		• All attributes except Letterhead Name	• Letterhead Name
List View		• All attributes except View Unique Name	• View Unique Name
Page Layout		• All attributes except Page Layout Name	• Page Layout Name
Record Type	• Description • Record Type Label	• Active • Business Process	• Name
Report		• All attributes except Report Unique Name	• Report 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	• Mobile Ready • Tab Style	• Tab Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
	• Width		
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 • Recipients	• Description
Workflow Field Update	• Description • Field Value • Formula Value	• Lookup • Notify Assignee	• 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 • Notify Assignee • Priority • Record Type	• Subject

Component	Developer Editable	Subscriber and Developer Editable	Locked
		• Status	

Component Behavior in Packages

When building an app for distribution, it is important to consider how packaging affects your app and its components. This section contains guidelines and information on special behaviors in packages. Use these guidelines to help you determine what to include in your packages, how to design your app, and how to distribute your packages (managed or unmanaged).

Note:

- For more information on the properties of each component in packages, see the packaged components properties table on page 5. For more information on the attributes of each component in packages, see the component attributes table on page 9.
- Component names must be unique within an organization. To ensure that your component names do not conflict with those in an installer's organization, use a managed package so that all of your component names contain your namespace prefix.

Analytic Snapshot

Developers of managed packages must consider the implications of introducing analytic 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 analytic snapshot referencing the report will not be installed, even though the Package Installation page may indicate that it will be. Also, if the subscriber has modified the report, that report may return results impacting the information displayed by the analytic snapshot. As a best practice, the developer should release an analytic snapshot and the related reports in the same version.

Custom objects are developer-controlled so analytic snapshot field mappings could become invalid when the package is upgraded. For example, if the developer deletes a custom field included in the field mapping, it would become invalid. Analytic snapshots can be created, installed, and run with invalid mappings as long as one valid mapping exists.

Since the running user is selected by the subscriber, some analytic snapshot field mappings could become invalid if the running user does not have access to source or target fields.

Apex Classes or Triggers

Any Apex that is included as part of a Force.com AppExchange 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 organization. The installer can decide whether or not to install the package if any tests fail.

Tip: To prevent naming conflicts, Salesforce recommends using managed packages for all packages that contain Apex. This way, all of the Apex objects contain your namespace prefix. For example, if there is an Apex class called `MyHelloWorld` and the namespace for your organization is `OneTruCode`, the class is referenced as `OneTruCode:MyHelloWorld`.

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

- If you are 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, you must explicitly package the individual languages desired in order for those translations to be included in the package.
- If you reference a custom object's sharing object (such as MyCustomObject__share) in Apex, this adds a sharing model dependency to your package. You must set the organization-wide sharing default access level for the custom

object to Private in order for other organizations to install your package successfully. For more information on setting the sharing model, see "Managing the Sharing Settings" in the Salesforce online help.
- The code contained in an Apex script that is part of a managed package is automatically obfuscated and cannot be viewed in an installing organization. The only exceptions are methods declared as global, meaning that the method signatures can be viewed in an installing organization.
- You can use the `deprecated` annotation in Apex to identify methods, classes, exceptions, enums, interfaces, and variables that can no longer be referenced in subsequent releases of the managed package in which they reside. This is useful when you are refactoring 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 cannot see the deprecated elements, while the elements continue to function for existing subscribers and API integrations.

Custom Fields

- Picklist field values for custom fields can be added, edited, or deleted by subscribers. A developer should carefully consider this when explicitly referencing a picklist value in code. Picklist values can be added or deleted by the developer. During a package upgrade, no new picklist values are installed into the subscriber's organization for existing fields. Any picklist values deleted by the developer are still available in the subscriber's organization.
- Auto-number type fields and required fields cannot be added after the object is uploaded in a managed-released package.
- Developers can add universally required custom fields to managed packages as long as they have default values.

Custom Labels

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

Custom Objects

- If a developer enables the `Allow Reports` or `Allow Activities` attributes on a packaged custom object, the subscriber's organization also has these features enabled during an upgrade. Once enabled in a managed-released package, the developer and the subscriber cannot disable these attributes.
- Standard button and link overrides are not packageable. The subscriber must manually configure these settings after installation.

Custom Report Types

A developer can edit a custom report type in a managed package after it is released. Subscribers automatically receive these changes when they install a new version of the managed package. However, developers cannot remove objects or fields from the report type once the package is released.

Custom Tabs

- The `Tab Style` for a custom tab must be unique within your app. However, it does not need to be unique within the organization where it is installed. A custom tab's style will not conflict with an existing custom tab in the installer's environment.
- If you want to provide custom tab names in different languages, see "Renaming Tab and Field Labels" in the Salesforce online help. Subscribers cannot 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 install. Also, if the subscriber has modified the report, that report may return results impacting what information is displayed by the dashboard component. As a best practice, the developer should 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 organization must have the divisions feature enabled to install the package.
- Setting the division filter on a report does not cause a dependency. The setting is dropped when installed into the subscriber's organization.
- 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 organization does not support divisions on the object, then the column is dropped during install.
- If you install a custom report type that includes an object's division field as a column, that column is dropped if the organization does not support divisions.

Field Dependencies

- Developers and subscribers can add, change, or remove field dependencies.
- If the developer adds a field dependency, it is added during installation unless the subscriber has already specified a dependency for the same field.
- If a developer removes a dependency, this change is not reflected in the subscriber's organization 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 is not reflected in the subscriber's organization during an upgrade.

Folders

- Components that Salesforce stores in folders, such as documents, cannot 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 organization using the publisher's folder names. Give these folders names that indicate they are 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 are not 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 are not included in the package and do not 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 that contains your message and add it to your custom home page layout.

Home Page Layouts

Once 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.

List Views

List views associated with queues cannot 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 organization is not 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"—does not cause a dependency. The report generates an error when run in the installers organization if it does not support the currency.
- If an object's currency field in a report is included as a column and the subscriber's organization is not enabled for multiple currencies, that column is dropped during install.
- If a subscriber installs a custom report type that includes an object's currency field as a column, that column is dropped if the organization is not enabled for multiple currencies.

Page Layouts

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

Person Accounts

Packages referring to fields related to person accounts on the Account object—for example, `Is Person Account`, `First Name`, `Last Name`, `Title`—cannot be uploaded.

Picklist Values

- Subscribers can rename or delete picklist field values. A developer should carefully consider this when explicitly referencing a picklist field value in Apex.
- Picklist field values can be added or deleted in the developer's organization. Upon upgrade, no new values are installed. Any values deleted by the developer are still available in the subscriber's organization until the subscriber deletes them.

Profile Settings

Once a managed package is  Managed - Released, the developer can make changes to the profile settings but subscribers will not get these changes when upgrading. Salesforce recommends contacting subscribers to ask them to make important profile changes manually when they upgrade.

Record Types

- If record types are included in the package, the subscriber's organization must support record types to install the package.
- When a new picklist value is installed, it is 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 organization is not using record types on the object or does not support record types, then the column is dropped during install.
- If you install a custom report type that includes an object's record type field as a column, that column is dropped if the organization does not support record types or the object does not have any record types defined.
- Record types on standard objects cannot be packaged and therefore should not be referenced by any components included in the package.

Reports

If a report includes elements that cannot be packaged, those elements will be dropped or the package upload will fail. 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 advanced filter conditions 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 have been moved to a private folder or to the Unfiled Public Reports folder.

Translation Workbench

- If you have 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 does not need to enable anything for the packaged language translations to appear. The only reason installers may want to enable the translation workbench is to change translations for unmanaged components after installation, override custom label translations in a managed package, or to translate into additional languages.
- If you are designing a package extension, you can include translations for the extension components but not additional translations for components in the base package.
- On install, a subscriber must activate a language if it is not active in the subscriber's organization.

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 all associated workflow actions, such as alerts and field updates. However, any time-based triggers are not included in the package. Notify your installers to set up any time-based triggers that are essential to your app.
- Workflow alerts, field updates, outbound messages, and tasks can be protected by the developer. For more information on protected components, see "Protected Components" in the Salesforce online help.
- Developers can associate or disassociate workflow actions with a workflow rule at any time. These changes, including disassociation, are reflected in the subscriber's organization upon install. In managed packages, a subscriber cannot 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, sales team, or case team roles may not be uploaded.
- On install, all workflow rules are inactive. The subscriber must deploy the package or individually activate the workflow rules.

Component Availability After Deployment

Many components have an attribute that controls whether they are available for end users. After installation, certain components are immediately available if they were available in the developer's organization. Other components are not available by default and must be activated using the package deploy process or they can be activated individually.

The following components are available by default based on the developer's setting.

- Apex triggers
- Email templates
- Folders

Note: Folders shared with specific users or groups are installed with this option selected, however, no users or groups are selected in the subscriber's organization.

- Letterheads
- Record types
- Validation rules

The following components are not available by default, and must be activated using the package deploy process or they can be activated individually. For more information, see "Deploying Installed Packages" in the Salesforce online help.

- Custom objects
- Custom report types
- Workflow rules

Versioning and Deprecation in Package Versions

A package version is a set of components uploaded in a package. It is denoted by a version number, `majorNumber.minorNumber`, such as 2.1. 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.

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.

Versioning and Deprecation Apex in Package Versions

Package developers can use the `deprecated` annotation in Apex to identify methods, classes, exceptions, enums, interfaces, and variables that can no longer be referenced in subsequent releases of the managed package in which they reside. This is useful when you are refactoring 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 cannot see the deprecated elements, while the elements continue to function for existing subscribers and API integrations. A deprecated identifier can still be referenced internally by the package developer.

Note: You cannot use the `deprecated` annotation in Apex classes or triggers in unmanaged packages.

Package developers can use Managed - Beta package versions for evaluation and feedback with a pilot set of users in different Salesforce organizations. If a developer deprecates an Apex identifier and then uploads a version of the package as Managed - Beta, subscribers that install the package version still see the deprecated identifier in that package version. If the package developer subsequently uploads a Managed - Released package version, subscribers will no longer see the deprecated identifier in the package version after they install it.

Behavior in Package Versions

A package component can exhibit different behavior in different package versions. This behavior versioning allows you to add new components to your package and refine your existing components, while still ensuring that your code continues to work seamlessly for existing subscribers. If a package developer adds a new component to a package and uploads a new package version, the new component is available to subscribers that install the new package version.

Versioning Apex Code Behavior

Apex has special syntax so that package developers can use conditional logic in 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 subscribers install multiple versions of your package and write code that references Apex classes or triggers in your package, they must specify the version that they are referencing. Within the Apex code that is being referenced in your package, you can conditionally execute different code paths based on the version setting of the calling Apex code that is making the reference. The package version setting of the calling code can be determined within the package code by accessing the `Package.Version.Request` object. Package developers can use this object and accompanying methods to determine the request context and exhibit different behavior for different versions of the package.

The following sample shows different behavior in a trigger for different package versions

```
trigger oppValidation on Opportunity (before insert, before update) {

    for(Opportunity o : Trigger.new){

        // New validation added in package version 1.5
        // Applies to all version of the managed package, except 1.0
        If(Package.Version.Request.isGreaterThan(Package.Version.1.0)){
            If(o.Probability >= 50 && o.Description == NULL){
                o.addError('All deals over 50% require a description');
            }
        }

        // Validation applies to all versions of the managed package
        If(o.IsWon == true && o.LeadSource == NULL){
            o.addError('A lead source must be provided for all Closed Won deals');
        }
    }
}
```

For more information about `Package.Version.Request`, see the *Force.com Apex Code Developer's Guide*.

Deprecation Effects for Subscribers

Package developers can use the `deprecated` annotation to identify methods, classes, exceptions, enums, interfaces, and variables that can no longer be referenced in subsequent releases of the managed package in which they reside. This section demonstrates how deprecation of an Apex method affects subscribers that install the managed package. The table shows a typical sequence of actions by a package developer in the first column and actions by a subscriber in the second column. Each row in the table denotes either a package developer or subscriber action.

Package Developer Action	Subscriber Action	Notes
Create a global Apex class, `PackageDevClass`, containing a global method `m1`.		
Upload as Managed - Released version 1.0 of a package that contains `PackageDevClass`.		
	Install version 1.0 of the package.	The `Version Number` for the package is 1.0. The `First Installed Version Number` is 1.0.
	Create an Apex class, `SubscriberClass`, that references `m1` in `PackageDevClass`.	

Package Developer Action	Subscriber Action	Notes
Deprecate `m1` and create a new method, `m2`.		
Upload as Managed - Released version 2.0 of the package.		
	Install version 2.0 of the package.	The `Version Number` for the package is 2.0. The `First Installed Version Number` is still 1.0. `SubscriberClass` still references version 1.0 of the package and continues to function, as before.
	Edit the version settings for `SubscriberClass` to reference version 2.0 of the package. Save the class. Note an error message indicating that `m1` cannot be referenced in version 2.0 of the package.	
	Change `SubscriberClass` to reference `m2` instead of `m1`. Successfully save the class.	

Salesforce Editions and Limits

Use the table below to determine the features and maximum number of items supported in each Salesforce Edition. The following sections provide more detailed information about their respective areas:

- Custom Field Limit Details on page 24
- Rules Limit Details on page 24

Feature	Personal Edition	Group Edition	Professional Edition	Enterprise Edition	Unlimited Edition	Developer Edition
Custom Apps	Not Available	1	5	10	Unlimited	10
Custom Objects	Not Available	50	50	200	2,000	400
Custom Report Types (Limits apply to all custom report types regardless of development status.)	Not Available	Not Available	10	30	100	10
Tabs	Not Available	5	10	25	Unlimited	100
Custom Fields (Additional restrictions apply for activities, long text area fields, relationship fields, and roll-up summary fields. For more information, see Custom Field Limit Details on page 24.)	5 per object	100 per object	100 per object	500 per object	500 per object	500 per object
Active Rules (Limits apply to any combination of *active* workflow, assignment, auto-response, and escalation rules. For more information, see Rules Limit Details on page 24.)	Not Available	Not Available	50 per object	50 per object	50 per object	50 per object
Formulas: Number of Relationships	5 per object	5 per object	5 per object	5 per object	5 per object	5 per object
Formulas: VLOOKUP Functions	10 per object	10 per object	10 per object	10 per object	10 per object	10 per object
Active Validation Rules	Not Available	20 per object	20 per object	100 per object	500 per object	100 per object
Sharing Rules	Not Available	Not Available	100	100	100	100

Feature	Personal Edition	Group Edition	Professional Edition	Enterprise Edition	Unlimited Edition	Developer Edition
Static Resources	Not Available	A single static resource can be up to 5 MB in size. An organization can have up to 250 MB of static resources, total.	A single static resource can be up to 5 MB in size. An organization can have up to 250 MB of static resources, total.	A single static resource can be up to 5 MB in size. An organization can have up to 250 MB of static resources, total.	A single static resource can be up to 5 MB in size. An organization can have up to 250 MB of static resources, total.	A single static resource can be up to 5 MB in size. An organization can have up to 250 MB of static resources, total.

Custom Field Limit Details

The number of custom fields allowed per object varies according to Salesforce edition. The number of activities, long text area fields, relationship fields, and roll-up summary fields varies as well. The following table shows the maximum limits for each custom field type with respect to edition.

Custom Field Limits	Personal Edition	Group Edition	Professional Edition	Enterprise Edition	Developer Edition	Unlimited Edition
Total Custom Fields Allowed	5	100	100	500	500	500
Activities	No additional limit	No additional limit	20	100	100	100
Long Text Area Fields	No additional limit	5	5	25	25	25
Relationship Fields	No additional limit	25	25	25	25	25
Roll-up Summary Fields	No additional limit	10	10	10	10	10

Rules Limit Details

Different types of rules can have restrictions, depending on your Salesforce edition. The following table shows the restrictions with respect to edition.

Restrictions	Personal Edition	Group Edition	Professional Edition	Enterprise Edition	Developer Edition	Unlimited Edition
Total Rules Allowed (Limits apply to any combination of workflow,	Not Available	Not Available	300 per object 500 per org	300 per object 500 per org	300 per object 500 per org	300 per object 500 per org

Restrictions	Personal Edition	Group Edition	Professional Edition	Enterprise Edition	Developer Edition	Unlimited Edition
assignment, auto-response, and escalation rules, both *active* and *inactive*.)						
Workflow Rules and Approval Processes	Not Available	Not Available	Not Available	For Enterprise, Developer, and Unlimited Editions, each workflow rule and approval process can have: • 10 time triggers • 40 immediate actions • 40 time-dependent actions per time trigger Note that for both immediate and time-dependent actions, there can be no more than: • 10 email alerts • 10 tasks • 10 field updates • 10 outbound messages		
Assignment, Auto-response, and Escalation Rules	Not Available	Not Available	For Professional, Enterprise, Developer, and Unlimited Editions, each assignment, auto-response, and escalation rule can have: • 3000 rule entries • 300 formula criteria rule entries • 25 filter criteria per rule entry			
Workflow Time Triggers Per Hour (For more information, see "Time-Dependent Action and Time Trigger Considerations" in the Salesforce online help.)	Not Available	Not Available	Not Available	500	50	1,000

Packaging Glossary

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

Review the following terminology with Force.com AppExchange:

App

A collection of components such as tabs, reports, dashboards, and Visualforce pages that address a specific business need. Short for "application." Salesforce provides standard apps such as Sales and Call Center. You can customize the standard apps to match the way you work. In addition, you can package an app and upload it to Force.comAppExchange 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 AppExchange.

AppExchange Directory

A Web directory where hundreds of apps are available to Salesforce customers to review, demo, comment upon, and/or install. Developers can submit their apps for listing on AppExchange if they wish to share them with the community.

Beta, Managed Package

In the context of managed packages, an early version of a managed package that is uploaded to Force.com AppExchange for 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.

Deprecated Component

A publisher may decide to refine the functionality in a managed package over time as the requirements evolve. This may involve redesigning some of the custom objects or fields in the managed package. Publishers cannot delete some components in a Managed - Released package, but they can deprecate a component in a later package version so that new subscribers do not receive the component, while the component continues to function for existing subscribers and API integrations. A publisher can deprecate Apex methods, classes, exceptions, enums, interfaces, or variables.

License Management Application (LMA)

A free AppExchange app that allows you to track sales leads and accounts for every user who downloads a managed package of yours from 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, or Developer Edition organization as your license management organization. For more information, go to sites.force.com/appexchange/publisherHome.

Managed Package

A collection of application components that are posted as a unit on AppExchange, and are associated with a namespace and a License Management Organization. A package must be managed for it to be published publicly on AppExchange, and for it to support upgrades. An organization can create a single managed package that can be downloaded and installed by many different organizations. They differ from unmanaged packages in that some components are locked, 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, so as 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. An extension requires that the base managed package be installed in the organization.

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.

Native App

An app that is built exclusively with setup (metadata) configuration on Force.com. Native apps do not require any external services or infrastructure.

Package

A group of Force.com 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 Force.com AppExchange together.

Package Dependency

Created when one component references another component, permission, or preference, which must exist 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 scripts

Permissions and preferences can include but are not limited to:

- Divisions
- Multicurrency
- Record types

Package Installation

Incorporates the contents of a package into your Salesforce organization. A package on 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 Publication

Publishing your package makes it publicly available on Force.com AppExchange. Apps can be found under specific categories and by doing a search for keywords.

Package Registration

Registering your package allows you to access information about users who visit or download your app from the Force.com AppExchange.

Package Version

A package version is a set of components uploaded in a package. It is denoted by a version number, `majorNumber.minorNumber`, such as 2.1. 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.

Private Sharing

The process of privately sharing a package uploaded to the AppExchange, by using the URL that you receive from Salesforce upon upload. This URL is not listed in the AppExchange. Using the unlisted URL allows you to share a package manually without making it public.

Publisher

The publisher of a package is the Salesforce user or organization that published the package on AppExchange. Publishers are required to register using their contact information after registering their package on AppExchange.

Subscriber

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

Test Drive

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

Unmanaged Package

An AppExchange package that cannot be upgraded or controlled by its developer. Unmanaged packages allow you to take any app components and move them "as is" to AppExchange without going through a lengthy publishing process.

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 sends it to Force.com AppExchange. Uploading your package is just the first step; all apps on AppExchange must be registered, making them privately available. The final and optional step is to publish it, making it publicly available on AppExchange.

Index