ISVforce Guide

Version 45.0, Spring '19

@salesforcedocs

Last updated: April 18, 2019

© Copyright 2000–2019 salesforce.com, inc. All rights reserved. Salesforce is a registered trademark of salesforce.com, inc., as are other names and marks. Other marks appearing herein may be trademarks of their respective owners.

CONTENTS

Chapter 1: Welcome to the ISVforce Guide . . . 1

Resources for Partners . . . 2
Roles in the Application Lifecycle . . . 2
How to Sign Up for Test Environments . . . 3

Chapter 2: ISVforce Quick Start . . . 4

Tutorial #1: Sign Up for AppExchange . . . 5
- Step 1: Sign Up for the Partner Program . . . 5
- Step 2: Create a Development and Test Environment . . . 5
- Step 3: Get a Business Org . . . 6
- Step 4: Edit Your Publisher Profile . . . 6
- Sign-up Summary . . . 6

Tutorial #2: Developing Your App . . . 7
- Step 1: Create an App . . . 7
- Step 2: Package Your App . . . 8
- Step 3: Assign a Namespace . . . 8
- Step 4: Upload a Beta . . . 9
- Step 5: Install and Test the Beta . . . 9
- Development Summary . . . 10

Tutorial #3: Publishing and Licensing . . . 11
- Step 1: Uploading to the AppExchange . . . 11
- Step 2: Create an AppExchange Listing . . . 11
- Step 3: Complete the AppExchange Listing . . . 12
- Step 4: Manage Licenses for Your App . . . 12
- Publishing and Licensing Summary . . . 13

Tutorial #4: Updating Your App . . . 13
- Step 1: Creating a Patch Organization . . . 14
- Step 2: Developing a Patch . . . 14
- Step 3: Uploading the Patch . . . 15
- Step 4: Installing or Pushing a Patch . . . 15
- Updating Your App Summary . . . 16

Chapter 3: Grow Your AppExchange Business . . . 17

AppExchange Trailblazer Scorecard . . . 18
AppExchange Trailblazer Scorecard Categories . . . 19
Create Your AppExchange Trailblazer Scorecard Strategies and Goals . . . 20
Get Started with the AppExchange Trailblazer Scorecard . . . 21
- View the AppExchange Trailblazer Scorecard . . . 21
- Grant Access to the AppExchange Trailblazer Scorecard . . . 22

Find Answers to AppExchange Trailblazer Scorecard Questions 22
AppExchange Trailblazer Scorecard FAQ 23
How often is the AppExchange Trailblazer Scorecard updated? 23
Why doesn't the AppExchange Trailblazer Scorecard show install or revenue growth for my listing? 23
Why doesn't the AppExchange Trailblazer Scorecard show installs for my listing? 23
Why doesn't the AppExchange Trailblazer Scorecard show my Trailhead badges and Salesforce certifications? 23
Can I use the AppExchange Trailblazer Scorecard if I'm a consulting partner? 23

Chapter 4: Design and Build Your Solution 24

Overview of Packages 25
Planning the Release of Managed Packages 26
Create a Package 27
Developing and Distributing Unmanaged Packages 28
Install Notifications for Unauthorized Managed Packages 28
Components Available in Managed Packages 29
Editing Components and Attributes After Installation 37
Components Automatically Added to Packages 44
Special Behavior of Components in Packages 46
Protected Components 56
Understanding Dependencies 56
Metadata Access in Apex Code 57
About Permission Sets and Profile Settings 58
Custom Profile Settings 59
Protecting Your Intellectual Property 60
Creating Packaged Applications with Chatter 60
Matching the Salesforce Look and Feel 61
Maintaining My Domain URLs 62
Developing App Documentation 62
About API and Dynamic Apex Access in Packages 63
Manage API and Dynamic Apex Access in Packages 65
Configuring Default Package Versions for API Calls 66
About the Partner WSDL 67
Generating an Enterprise WSDL with Managed Packages 68
Work with Services Outside of Salesforce 68
Architectural Considerations for Group and Professional Editions 69
Features in Group and Professional Editions 70
Limits for Group and Professional Editions 71
Access Control in Group and Professional Editions 71
Using Apex in Group and Professional Editions 71
API Access in Group and Professional Editions 72
Designing Your App to Support Multiple Editions 73
Sample Design Scenarios for Group and Professional Editions 75

Connected Apps . . . 76
Create a Connected App . . . 76
Edit, Reconfigure, or Delete a Connected App in Salesforce Classic . . . 85
Install a Connected App . . . 87
View Connected App Details . . . 88
Manage a Connected App . . . 89
Edit Connected App Behavior . . . 91
Monitor Usage for an OAuth Connected App . . . 95
Uninstall a Connected App . . . 97
Environment Hub . . . 98
Get Started with the Environment Hub . . . 99
Manage Orgs in the Environment Hub . . . 101
Single Sign-on in the Environment Hub . . . 103
Environment Hub Best Practices . . . 105
Environment Hub FAQ . . . 105
Considerations for the Environment Hub in Lightning Experience . . . 108
Developer Hub . . . 108
Scratch Org Allocations for Partners . . . 108
Enable Dev Hub in Your Org . . . 109
Add Salesforce DX Users . . . 110
Free Limited Access License . . . 111
Manage Scratch Orgs from Dev Hub . . . 111
Link a Namespace to a Dev Hub Org . . . 112
Supported Scratch Org Editions for Partners . . . 112
Notifications for Package Errors . . . 113
Set the Notification Email Address . . . 114

Chapter 5: Package and Test Your Solution . . . 115

About Managed Packages . . . 116
Configure Your Developer Settings . . . 116
Register a Namespace Prefix . . . 117
Specifying a License Management Organization . . . 118
Create and Upload a Managed Package . . . 120
View Package Details . . . 122
Installing a Package . . . 125
Component Availability After Deployment . . . 128
Uninstalling a Package . . . 128
Installing Managed Packages Using the API . . . 129
Resolving Apex Test Failures . . . 130
Running Apex on Package Install/Upgrade . . . 130
How does a Post Install Script Work? . . . 131
Example of a Post Install Script . . . 132
Specifying a Post Install Script . . . 134
Running Apex on Package Uninstall . . . 134

How does an Uninstall Script Work? . . . 134
Example of an Uninstall Script . . . 135
Specifying an Uninstall Script . . . 136
Publishing Extensions to Managed Packages . . . 136

Chapter 6: Pass the AppExchange Security Review . . . 137

Security Review . . . 138
Security Review Steps . . . 138
Security Review Wizard . . . 140
Submit a Client or Mobile App for Security Review . . . 141
Submit an Extension Package for Security Review . . . 141
Security Review Resources . . . 141
Security Review FAQ . . . 142

Chapter 7: Publish Your Solution on AppExchange . . . 146

What Is the AppExchange? . . . 147
Business Plans for AppExchange Listings . . . 147
Publish on the AppExchange . . . 148
Connect a Packaging Organization to the AppExchange . . . 149
Create or Edit Your Provider Profile . . . 149
Create or Edit an AppExchange Listing . . . 149
Add a Business Plan to an AppExchange Listing . . . 150
Make Your AppExchange Listing Effective . . . 150
Choose an Installation Option . . . 151
Register Your Package and Choose License Settings . . . 152
Submit Your Listing for Security Review . . . 152
Email Notifications . . . 153
AppExchange Checkout . . . 153
Pricing Models in AppExchange Checkout . . . 154
Payment Methods in AppExchange Checkout . . . 155
AppExchange Checkout Considerations . . . 155
Get Started with AppExchange Checkout . . . 156
AppExchange Checkout FAQ . . . 161
Checkout Management App . . . 163
Checkout Management App Best Practices . . . 164
Checkout Management App Objects . . . 165
Get Started with the Checkout Management App . . . 166
Sample Checkout Management App Customizations . . . 171
Update Settings in the Checkout Management App . . . 173
View Checkout Management App Logs . . . 174
Work with AppExchange Leads . . . 174
AppExchange Leads FAQ . . . 175
Analytics Reports for Publishers . . . 177
Update the Package in an AppExchange Listing . . . 178

AppExchange FAQ . . . 179
Can I have multiple listings for an app or component? . . . 183

Chapter 8: Manage Orders . . . 184

Channel Order App . . . 185
Channel Order App Objects . . . 185
Order Types . . . 187
Order Status . . . 188
Channel Order App Permission Sets . . . 189
Set Up the Channel Order App . . . 190
Install the Channel Order App . . . 190
Assign Permission Sets to Channel Order App Users . . . 191
Define a Channel Order App Email Service . . . 191
Connect the Channel Order App to Salesforce . . . 192
Display Customers in the Channel Order App . . . 193
Assign Page Layouts in the Channel Order App . . . 193
Upgrade the Channel Order App . . . 194
Channel Order App Upgrade Considerations . . . 194
Upgrade the Channel Order App . . . 195
Field Mapping in Channel Order App v2 and Later . . . 197
Manage Orders in the Channel Order App . . . 199
Submit an Order . . . 199
Edit an Order . . . 201
Clone an Order . . . 202
Recall an Order . . . 202
Delete a Draft Order . . . 202
Fix Errors on Returned Orders . . . 203
Channel Order Apex API . . . 203
CHANNEL_ORDERS Namespace . . . 203
Service Order . . . 218
Service Order Detail . . . 220
Partner Order Submit API . . . 223

Chapter 9: Manage Licenses . . . 226

License Management App . . . 227
How Does the License Management App Work? . . . 227
Integrate the License Management App into Your Business Processes . . . 231
Best Practices for the License Management App . . . 232
Get Started with the License Management App . . . 232
Install the License Management App . . . 232
Associate a Package with the License Management App . . . 233
Configure the License Management App . . . 234
Manage Leads and Licenses for Your Offering . . . 235
Modify a License Record in the License Management App . . . 235

Change the Lead Manager in the License Management App . . . 236
Refresh Licenses for an Offering in the License Management App . . . 236
Move the License Management App to Another Salesforce Org . . . 237
Troubleshoot the License Management App . . . 237
Leads and Licenses Aren't Being Created . . . 237
Proxy User Has Deactivated Message . . . 238
License Management App FAQ . . . 238
Is the LMA compatible with Lightning Experience? . . . 238
Can I install the LMA in a non-production Salesforce org? . . . 239
Why can't I see the Modify License button on my license records? . . . 239
A customer installed my package before I associated it with the LMA. How can I manage the license record? . . . 239
Can I automate the assignment of licenses to users in the subscriber org? . . . 239
Why aren't leads and licenses being created in the LMA? . . . 239
What happens when I decrease the number of available licenses below the current number of licensed users? . . . 239

Chapter 10: Manage Features . . . 240

Feature Parameter Metadata Types and Custom Objects . . . 241
Set Up Feature Parameters . . . 242
Install and Set Up the Feature Management App in Your License Management Org . . . 242
Create Feature Parameters in Your Packaging Org . . . 243
Add Feature Parameters to Your Managed Package . . . 243
Reference Feature Parameters to Drive App Behavior and Track Activation Metrics . . . 244
How Do Feature Parameters Work? . . . 244
Drive App Behavior with LMO-to-Subscriber Feature Parameters . . . 244
Track Preferences and Activation Metrics with Subscriber-to-LMO Feature Parameters . . . 245
Hide Custom Objects and Custom Permissions in Your Subscribers' Orgs . . . 246
Best Practices for Feature Management . . . 246
Considerations for Feature Management . . . 247

Chapter 11: Provide a Free Trial of Your Solution . . . 248

Why Use Trialforce? . . . 249
Trialforce . . . 249
Set Up Trialforce . . . 251
Link a Package with Your License Management Organization . . . 251
Request a Trialforce Management Org . . . 252
Setting Up Custom Branding for Trialforce . . . 252
Create a Trialforce Source Organization . . . 254
Create a Trialforce Template . . . 255
Link a Trialforce Template to the AppExchange . . . 256
Submit a Trialforce Template for Security Review . . . 256
Provide a Free Trial on the AppExchange . . . 256
Provide a Free Trial on the AppExchange Using Trialforce . . . 257

Offer a Test Drive on AppExchange . . . 257
Provide a Free Trial on the AppExchange When Your Offering Is Installed . . . 258
Provide Free Trials on Your Website . . . 258
Enable the SignupRequest API . . . 258
Choose a Sign-Up Form Hosting Option . . . 259
Create Sign-Ups Using the API . . . 260
Provision Trial Orgs . . . 260
Modify the Trial for an Upgrade . . . 260
Trialforce Best Practices . . . 261
Trialforce FAQ . . . 262

Chapter 12: Support Your AppExchange Customers . . . 264

Subscriber Support Console . . . 265
Viewing Subscriber Details . . . 265
Request Login Access from a Customer . . . 265
Logging In to Subscriber Orgs . . . 266
Troubleshooting in Subscriber Organizations . . . 266
Usage Metrics . . . 267
Setting up Usage Metrics . . . 267
Accessing Usage Metrics Data . . . 268
MetricsDataFile . . . 269
Usage Metrics Visualization . . . 271

Chapter 13: Update Your Solution . . . 274

About Package Versions . . . 275
Create and Upload Patches . . . 276
Working with Patch Versions . . . 277
Versioning Apex Code . . . 279
Apex Deprecation Effects for Subscribers . . . 280
Publish Upgrades to Managed Packages . . . 281
Delete Components in Managed Packages . . . 281
Viewing Deleted Components . . . 283
Modifying Custom Fields after a Package is Released . . . 285
Manage Versions . . . 285
Pushing an Upgrade . . . 285
Push Upgrades . . . 286
Push Upgrade Best Practices . . . 286
Assigning Access to New Components and Fields . . . 288
Sample Post Install Script for a Push Upgrade . . . 288
Scheduling Push Upgrades . . . 290
View Push Upgrade Details . . . 292
View an Organization's Upgrade History . . . 293

APPENDICES . . . 294

Appendix A: ISVforce User License Comparison 294

Appendix B: OEM User License Comparison 298

GLOSSARY 302

INDEX 305

CHAPTER 1 Welcome to the ISVforce Guide

In this chapter ...

- Resources for Partners
- Roles in the Application Lifecycle
- How to Sign Up for Test Environments

Hello, partner! Welcome to the ISVforce Guide. Learn to plan, build, distribute, market, sell, and support solutions that run on the Salesforce platform. Get your feet wet with the quick start tutorials. Then, dive in for a close look at the concepts, procedures, tools, and resources you need to successfully navigate the stages of the solution lifecycle.

Resources for Partners

The Partner Community, at https://partners.salesforce.com, is the primary resource for all ISVs. To get started, we recommend visiting the Education page, your one-stop shop for all ISV content. In addition, you can use the Partner Community to:

- Collaborate with other partners and salesforce.com using our Chatter community.
- Stay up-to-date on news and events related to the Salesforce Partner Program.
- Log cases for access to partner-specific features and customer support.
- Use enhanced search, integrated with the Trailblazer Community, to quickly find relevant resources.
- Browse the Salesforce Partner Online Training catalog and sign up for courses.

The Partner Community is self-service—the first person to register your partnership becomes your designated administrator and manages the creation of additional users for your company. You can change or add administrators, as required.

Roles in the Application Lifecycle

This guide covers the entire lifecycle of a package application, so some of the topics might not be relevant to you. The following list has topic suggestions by role.

An application *architect*

The application architect determines the scope of the application and the internal structures that support it. Architects need to know details about the underlying Lightning Platform platform that will determine not only the application's use, but which editions it supports, how it's installed, configured, and upgraded. Architects need to be familiar with this entire guide, but especially the following chapters:

- Design and Build Your Solution on page 24
- Pass the AppExchange Security Review on page 137

A *developer* creates, packages, and uploads an app

A developer, or often a team of developers, create an app, package it, and upload it to the AppExchange. Developers also update the app with bug fixes and new features. As a developer, you'll want to see the following chapters:

- Design and Build Your Solution on page 24
- Package and Test Your Solution on page 115
- Developing App Documentation on page 62
- Update Your Solution on page 274

A *publisher* distributes, sells, and supports the app

The publisher of an app is the person or company who has a profile and one or more listings for the app on AppExchange. Publisher listings contain a link to an app they have uploaded to AppExchange, or to a third-party website. Publishers also set default license settings. As a publisher, you'll want to see the following chapters:

- Publish Your Solution on AppExchange on page 146
- Provide a Free Trial of Your Solution on page 248
- Support Your AppExchange Customers on page 264

An *administrator* installs the app

An administrator, or *admin*, downloads your app from AppExchange and installs it into their organization. Admins might also customize the app to further suit their business needs. See the following topic to learn how admins will interact with your app.

- Installing a Package on page 125

How to Sign Up for Test Environments

To sign up for test environments (organizations), use the Environment Hub.

Note: If you're a new Salesforce user, log in to the organization that you received when you signed up for the Partner Program. The Environment Hub is enabled in this organization by default. If you're an existing Salesforce user and are using a different organization to manage development, log a case in the Partner Community to enable the Environment Hub.

1. Log in to the organization where Environment Hub is enabled.
2. Select the Environment Hub tab, then click **Create Organization**.
3. In the Purpose drop-down list, select **Test/Demo**.
4. In the Edition drop-down list, choose the edition you want to test against.
5. Fill in the remaining required fields. Optionally, set up My Domain.
6. Agree to the terms and then click **Create**.
7. You'll receive an email that will prompt you to log in and change your password. Click the link, change your password, and create a password question and answer.

CHAPTER 2 ISVforce Quick Start

In this chapter ...

- Tutorial #1: Sign Up for AppExchange
- Tutorial #2: Developing Your App
- Tutorial #3: Publishing and Licensing
- Tutorial #4: Updating Your App

Get ready to build and sell solutions on AppExchange by completing short tutorials.

Note: Some features in this quick start are available only to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, visit https://partners.salesforce.com.

Tell Me More

If you're new to app development, or if you're a smell-the-roses type, review these resources before you get started.

- For an introduction to the Salesforce platform, complete the Admin Beginner, Admin Intermediate, and Developer Beginner trails.
- For Salesforce developer documentation and other resources, visit Salesforce Developers at https://developer.salesforce.com.
- For a list of useful terms, see the Glossary on page 302.

Tutorial #1: Sign Up for AppExchange

In this tutorial, you set up the tools you need to develop, sell, and support apps and components built on the Lightning Platform platform. You start by signing up for the Partner Program. You then have access to the Partner Community, which allows you to view helpful resources, create support cases, and collaborate with other partners and Salesforce. The Partner Community is also the best source for news and events about the Partner Program. In addition, you can access the Environment Hub, where you can create development and test organizations.

If you're familiar with Salesforce, you know that an organization is a cloud unto itself. If you're new to Salesforce, think of your organization as a separate environment for developing, testing, and publishing your offering.

Step 1: Sign Up for the Partner Program

The first step is to sign up for the Partner Program.

1. In your browser, go to https://partners.salesforce.com and click **Join Now**.

   Note: The signup process varies according to the region or country. Follow the instructions presented.

2. Fill in the fields about you and your company.
3. Select the first option: **Independent Software Vendor (ISV)**.
4. Click **Submit Registration**.

In a moment, you'll receive a confirmation, followed by an email welcoming you to the Partner Program and including login credentials.

Congratulations, you're now part of the Salesforce ISV Partner Program! Click the link to the Partner Community (https://partners.salesforce.com) and log in. Bookmark this page. You'll be using it a lot.

Step 2: Create a Development and Test Environment

To build and sell on the Lightning Platform platform, you need different environments for different tasks. We call these environments *organizations*, or orgs for short. You use the Environment Hub to create these orgs. The first org you need is the Partner Developer Edition, which is where you develop and package your offering. If you already have a Developer Edition org, we recommend signing up for the Partner Developer Edition org because you can have more data storage, licenses, and users.

Note: If you're a new Salesforce user, log in to the organization that you received when you signed up for the Partner Program. The Environment Hub is enabled in this organization by default. If you're an existing Salesforce user and are using a different organization to manage development, log a case in the Partner Community to enable the Environment Hub.

1. Log in to the organization where the Environment Hub is enabled, usually your partner business org.
2. Click the **Environment Hub** tab, and then click **Create Organization**.
3. In the Purpose drop-down list, select **Development**. For simplicity, we refer to this as your *dev org*.
4. Fill in the required fields. Optionally, set up My Domain.
5. Agree to the terms and then click **Create**.
6. In the Purpose drop-down list, select **Test/Demo** and **Partner Enterprise** for the org edition. This process creates a test org, where you test the app or component that your are developing.
7. Shortly, you'll receive emails that prompt you to log in and change your password for your dev and test orgs.

Tell Me More...

The Environment Hub has several types of test orgs available, because different editions of Salesforce have different features. If you plan to distribute your app or component to a particular edition, you want to test your offering and make sure that it works there. Although that's beyond the scope of this quick start. For more information, see Architectural Considerations for Group and Professional Editions on page 69.

Step 3: Get a Business Org

In the previous step, you created orgs for developing and testing your offering. To manage sales and distribution, you need one more org. In this step, you log a case in the Partner Community to have a partner business org provisioned for you. Your Partner Business Org contains the apps that you use to manage sales and distribution, including the License Management App (LMA) and Channel Order App (COA).

1. In the Partner Community, under the Support tab, select **New Case**.
2. Select **Request Partner Benefits**, and then select **Create a Case**.
3. In the Description field, tell us if you have an existing org or if you need a new one. If you have an existing Salesforce org, enter the Org ID in the Description field to add two more CRM licenses to your org. If you don't have an existing org, we provide a new one for you. In either case, make sure to enter your business address and then select **Submit Case**.

   Note: It can take 24–48 hours for your case to be closed. You can check the status of your case at any time under the Support tab of the Partner Community.

4. You'll receive an email prompting you to log in and change your password. Do that, and then bookmark the page.

Step 4: Edit Your Publisher Profile

In this step, you log in to the Partner Community and provide information about your company. We display some of this information on AppExchange listings to help customers get to your business.

1. Log in to the Partner Community using the username and password of your business org.
2. On the Publishing page, click **Company Info**.
3. Fill out the information in the Provider Profile, and then click **Save**.

Sign-up Summary

In this first tutorial, you signed up for the Partner Program and all the organizations you need to develop, test, and sell your offering. Let's review what you signed up for and the purpose of each.

Partner Program
: The Partner Program gives you access to the Partner Community, where you can get help and training information, log cases for support issues, and collaborate with other partners. You also get access to the Environment Hub, which lets you create and manage new test and development orgs.

Partner Developer Edition
: Also known as your *dev org*, this is where you develop your offering and eventually package it for distribution.

Test Organization
: Also know as your *test org*, this is where you install and test your offering.

Partner Business Organization
: This is where you license and manage your offering.

Tutorial #2: Developing Your App

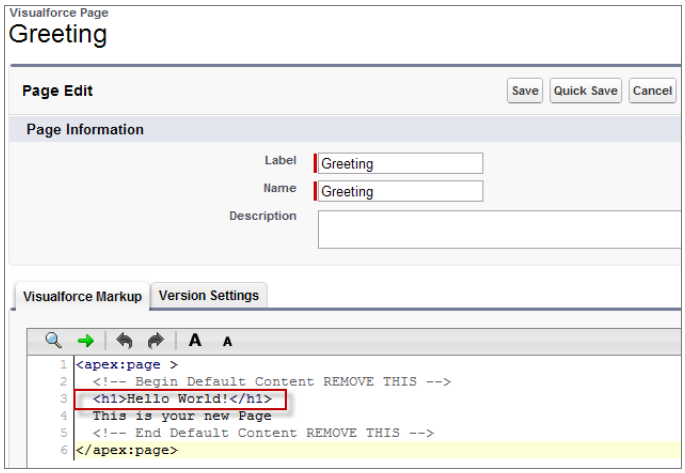
In this tutorial you'll create a very simple "Hello World" application. It won't do much, but it's enough to understand where development takes place in the lifecycle of a packaged application.

Step 1: Create an App

In this step you're going to create an app that contains a page, and a tab to display the page.

1. In your browser, log in to your Partner Developer Edition organization. Hereafter we'll call this your "dev org".
2. From Setup, enter *`Visualforce Pages`* in the `Quick Find` box, then select **Visualforce Pages**.
3. In the Visualforce list, click **New**.
4. In the `Label` field enter *`Greeting`*.
5. In the Visualforce Markup area, replace the contents of the `<h1>` tag with *`Hello World`*.

Visualforce Page Editor



6. Click **Save**.

Now you'll associate the page with a tab.

1. In the sidebar menu, enter *`Tabs`* in the `Quick Find` box, then select **Tabs**.
2. In the Visualforce Tabs list, click **New**.
3. In the New Visualforce Tab wizard, click the drop-down box and select the Hello World page you just created.
4. For the Tab Label, enter *`Hello`*.
5. Click the Tab Style field and choose any icon to represent your tab.
6. Click **Next**, then **Next** again, and **Save** on the final page.

Now you'll create a new app that contains your tab and page.

1. In the sidebar menu, enter *`Apps`* in the `Quick Find` box, then select **Apps**.
2. Click **New**.
3. In the App Label field enter Hello World and then click **Next** and **Next** again on the following page.

4. On the Choose the Tabs page, scroll to the bottom of the Available Tabs list, find your Hello tab, and add it to the Selected Tabs list. Click **Next**.
5. Select the **Visible** checkbox to make this app visible to all profiles and then click **Save**.

Tell Me More....

If it seems like you just created a page within a container, within another container, you did. And you're about to put all of that in another container! What's with all these containers and what do they do?

- A *tab* is a container for things you want to display on the same page, such as a chart, a table, or the Visualforce page your created.
- An *app* is a container for tabs that appear next to each other. When you create an app, it's available in the app picker in the upper right hand corner of the screen.
- A *package* is a container for things you upload to the AppExchange. Usually a package contains an app your customers can install in their org, but you can also upload packages that extend existing apps. You haven't created a package yet, you'll do that in the next step.

Step 2: Package Your App

In this step you'll package the app so you can distribute it on the AppExchange. A package is simply a container for components. In this case it's your app, tab, and page.

1. From Setup, enter `Packages` in the `Quick Find` box, select **Packages**, and then click **New**.
2. In the Package Name field enter `Hello World` and then click **Save**.
3. On the Package Detail page click **Add Components**.
4. Select your Hello World app and then click **Add to Package**.

Tell Me More....

When you clicked **Add to Package**, did you notice that your Hello tab and Greeting page were automatically added to the package? When you create a package, the framework automatically detects dependent components and adds them to the package.

Step 3: Assign a Namespace

In this step you'll choose a unique identifier called a namespace. A namespace differentiates your components from other components and allows you to do things such as upgrade the app after it's been installed. Choose your namespace carefully as it can't be changed later.

1. From Setup, enter `Packages` in the `Quick Find` box, then select **Packages**.
2. In the Developer Settings list, click **Edit** and on the following page click **Continue**.
3. In the Namespace Prefix field, enter a 1-15 character alphanumeric ID and then click **Check Availability**. Repeat this step until you have a unique namespace.
4. In the `Package to be managed` field choose your Hello World package and then click **Review Your Selections**.
5. Review the information on the page and then click **Save**.

Tell Me More....

Within the underlying code, your namespace is prepended to all components that are packaged from your dev org. This allows your package and its contents to be distinguished from those of other developers, and ensures your exclusive control of all packaged components.

Step 4: Upload a Beta

Before you upload a production version of your app, it's a common practice to upload a beta version for testing.

1. From Setup, enter `Packages` in the `Quick Find` box, then select **Packages**.
2. On the Packages page, click your **Hello World** package and then click **Upload**.
3. On the Upload Package page, enter a version name and number.
4. For the Release Type, make sure to choose **Managed — Beta**.
5. Scroll to the bottom and click **Upload**. It may take a moment for the upload to complete.

Congratulations, you've uploaded an app to the AppExchange! Your app isn't available to the public, but you can access it through an install link. You'll do that in the next step.

Tell Me More....

The purpose of a beta is for testing only. Therefore, a beta can only be installed in a test org, Developer Edition, or sandbox (more on that later). Next you'll install the beta in the test org you created in Step 2: Create a Development and Test Environment.

Step 5: Install and Test the Beta

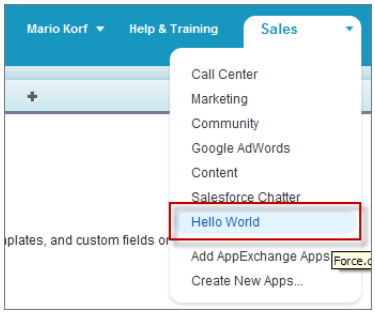
Installing the beta is easy, just click the link and provide the username and password you use for your test org.

1. Click the Installation URL now.

Installation URL Link

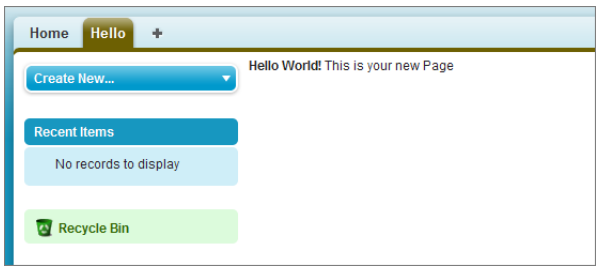
2. On the login page, enter the Username and Password of your test org.
3. On the Package Installation Details page, click **Continue**.
4. Click **Next**.
5. On the Security Level page, **Grant access to all users** and click **Next**.
6. Click **Install**.
7. Once the installation completes, you can select your app from the app picker in the upper right corner.

Hello World App



8. You should see your Hello tab, and the greeting text on your page.

Hello World Tab and Page



At this point you would normally test the application and make sure it works as designed. Your app installs easily and displays what you want, so let's move on.

Tell Me More....

Beta packages can also be installed in sandboxes. A sandbox is a replica of your customer's org that allows them to develop, test, or install apps, and verify the changes they want to commit. None of the orgs you've signed up for in this workbook have a sandbox, but if you have a sandbox in another org and want to install your app in it, you must replace the initial portion of the **Installation URL** with `http://test.salesforce.com`.

Development Summary

Congratulations, you just completed an essential part of the software development lifecycle! Further changes to your app will follow the same procedure:

1. Modify the existing app in your dev org.
2. Package the app.
3. Upload as a beta package.
4. Install the beta in a test org.
5. Test the installed app.

Tutorial #3: Publishing and Licensing

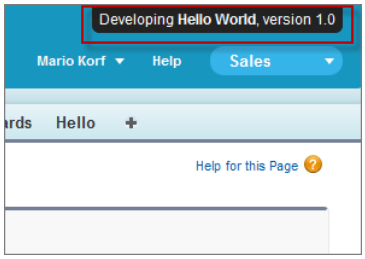
Imagine you've been through a few development cycles with your beta and you're ready to publish a public app. The next step is to upload a production app, or what we call a *managed released* version of your app. Then you can create a listing so that other people can find your app and know what it does. Finally, you want to connect your app to your business org so you manage the licenses for people that install your app.

Step 1: Uploading to the AppExchange

This step will seem familiar, it's similar to uploading a beta.

1. If you've been following along non-stop, you're probably still logged in to your test org. Go ahead and log in to your dev org now.
2. Notice in the upper right corner there's a link that says **Developing Hello World, version 1.0**. Click that link to go directly to the Package Detail page.

Developing Hello World, version 1.0

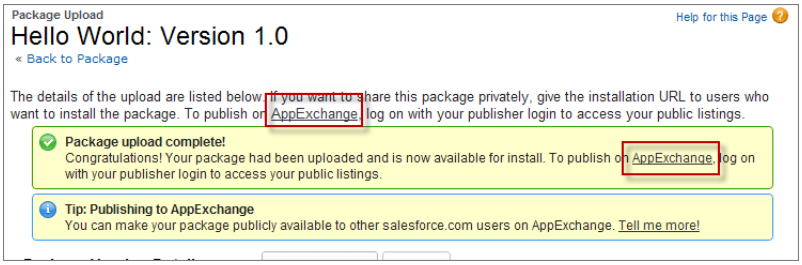


3. On the Package Detail page, click **Upload**.
4. For the Release Type, choose **Managed — Released**.
5. Scroll to the bottom and click **Upload**.
6. Click **OK** on the popup.

Step 2: Create an AppExchange Listing

In this step, you create an AppExchange listing, which is the primary way customers discover apps, components, and services to enhance their Salesforce experience.

1. After your package uploads, click the link to publish on the AppExchange. You are directed to the Listings tab on the Publishing page.



2. If prompted, enter your login credentials for the Partner Community.

3. Read and agree to the terms and conditions, and then click **I Agree**.
4. The first question asks if you've already listed on the AppExchange. You did that in Tutorial 1, Step 4: Edit Your Publisher Profile on page 6, so select **Yes** and click **Continue**.
5. Click **Link New Organization**.
6. You're prompted for your username and password. Enter the values for your development org.
7. Click the **Publishing** tab.
8. Click **New Listing**.
9. Enter a listing title, such as `Hello World App by <your name>`. Adding your name helps ensure that your listing title is unique.
10. Choose **App**, and then click **Save & Next** to open the AppExchange publishing console.
11. On the Text tab, fill in the required fields, and then click **Save & Next** again.

Tell Me More...

Don't be concerned with making your listing perfect, because it's not public yet, and you can change the listing at any time.

Step 3: Complete the AppExchange Listing

Many customers like to see and experience a product before they decide to purchase. We give you several ways to show off your app or component in an AppExchange listing. For example, you can add screenshots and videos to draw attention to key features, or add white papers to help demonstrate business value. You can also let customers try your offering in their own organizations or set up a test environment that you've customized.

1. If you're not already there, click the Media tab in the AppExchange publishing console.
2. Add an app logo, tile image, and screenshot. Because your listing isn't used outside of this tutorial, use any image file that you have available.
3. Click the **App** tab, and then select **An app that includes a package (entirely or in part)**.
4. Click **Select Package** and choose the package that you uploaded in the previous step.
5. For the installation method, select **Directly from the AppExchange**.
6. Choose whether you want the app to be installed for every user in the customer's organization or just system administrators. For this tutorial, either option is fine.
7. For app specifications, select editions and languages. For this tutorial, you can select any available edition and language.
8. Click **Save & Next**.
9. Click **Save & Next** twice, because you don't want to configure a free trial or set up lead collection for this app.
10. For pricing, select **Free**. Use the default values for all other fields.
11. Agree to the terms and conditions, and then click **Save**.

Congratulations—you've completed your first listing! Like everything else you've done so far, you can go back and change it later if you want.

Step 4: Manage Licenses for Your App

The License Management App (LMA) helps you manage sales, licensing, and support for your offering. The LMA comes preinstalled in your business organization. In this step, you connect your app to the LMA.

Note: This feature is available to eligible partners. For more information on the Partner Program, including eligibility requirements, visit www.salesforce.com/partners.

1. If you haven't done so already, log in to the Partner Community.
2. On the Publishing page, click the **Packages** tab.
3. Find the package that you want to link, and then click **Manage Licenses**.
4. Click **Register**.
5. Enter the login credentials of your partner business org, and then click **Submit**.
6. For the default license type, choose free trial.
7. Enter a trial length in days.
8. For the number of seats, choose the site-wide license.
9. Click **Save**.

It can take up to 30 minutes for your app to be connected to the LMA. Take a break; you've earned it!

Publishing and Licensing Summary

In this tutorial, you uploaded your managed-released app to the AppExchange and created a listing for your app. You also linked your app to the License Management App, available in your business organization. You can use the LMA to manage and renew licenses and to set default license settings. For example, you can license your app as a free trial that expires after a specified number of days. For more information, see Manage Licenses on page 226.

Right now your app has a private listing on the AppExchange that you can share with potential customers, but the public doesn't see it unless they have the link. Before you can list the app publicly, you'll need to pass a security review, which is beyond the scope of this quick start. For more information, see Security Review Steps on page 138.

Tutorial #4: Updating Your App

If you're familiar with Salesforce, you know we do weekly patch releases to fix bugs, and a few times a year we have a major release to introduce new features. As an ISV, you can do the same thing by delivering a patch release to fix bugs and a major release for new features.

- For new features, the process is the same as you've experienced. You start by modifying your app, package it, upload a beta, test the beta, and then upload a managed-released version. Major releases increment the version to the next whole number, from 1.0 to 2.0, for example, and minor releases to the first dot from 1.0 to 1.1. There are no hard rules for what constitutes a major or minor release. That's up to you.
- For bug fixes, the process is slightly different. You start by creating a patch org, a special environment which has limited functionality and can only be used to develop a patch for a specific package. After you upload the patch, you have the option of pushing the patch to your customers, so they get your bug fixes the next time they log in. Minor releases increment the version number to the second decimal, from 1.0 to 1.0.1, for example.
- Major or minor releases must be installed by customers (pulled). However, you can push patch releases directly to customer orgs. This feature is only available to registered ISVforce/OEM partners. For more information on the Partner Program, including eligibility requirements, please visit us at www.salesforce.com/partners.

Since the process for developing a major release is already familiar, let's do a patch release and then deliver it by pushing the patch to our customers.

Step 1: Creating a Patch Organization

In order to create a patch, you need to generate a new patch development organization.

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**.

   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.

In a moment you receive an email with your login credentials. After you log in and change your password, proceed to the next step.

Tell Me More

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.

Step 2: Developing a Patch

We're going to make a simple change to your app. Instead of displaying just Hello World, you'll add today's date.

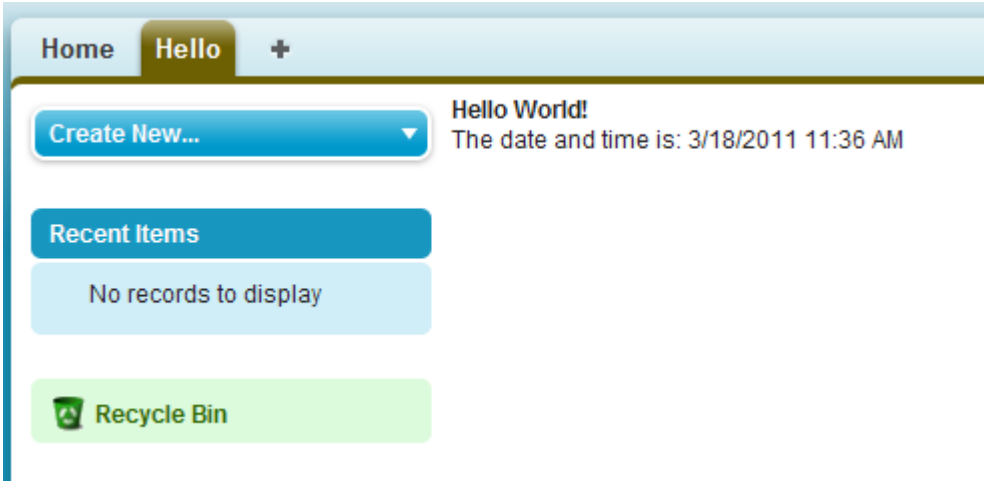
1. In your patch org, from Setup, enter `Packages` in the `Quick Find` box, select **Packages**, then click your **Hello World** package.
2. In the list of Package Components, click your **Greeting** page.
3. Click **Edit**.
4. Right after the closing `</h1>` tag, enter the following:

```
<br/>
<apex:outputText value="The date and time is: {!NOW()}"/>
```

5. Click **Save**.

6. To see the output, click the **Hello** tab and you'll notice that today's time and date are displayed.

Display the date and time



That's as much as we need to do in this patch. Let's move on.

Tell Me More....

The `!NOW` function returns the date in a standard format. There are many more built-in functions and ways to format the output. For more information, see the *Visualforce Developer's Guide*.

Step 3: Uploading the Patch

Typically the next step is to upload a beta patch and install that in a test organization. Since this is very similar to Step 4: Upload a Beta and Step 5: Install and Test the Beta, that you completed in Tutorial #2: Developing Your App, we won't make you do that again.

1. In your patch org, from Setup, enter *`Packages`* in the `Quick Find` box, select **Packages**, and click your **Hello World** package.
2. On the Upload Package page, click **Upload**.
3. Enter a version name, such as today's date.
4. Notice that the `Version Number` has had its `patchNumber` incremented.
5. Select **Managed — Released**.
6. 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.
7. 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.
8. Click **Upload**.

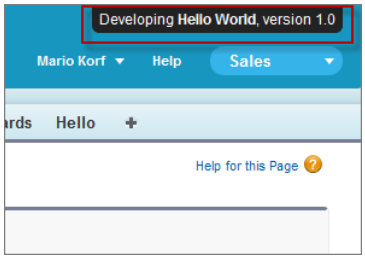
Congratulations, you've uploaded a patch release. You'll want to share that patch with others, and you'll do that next.

Step 4: Installing or Pushing a Patch

There are two ways to deliver a patch, you can have your customers install it, or you can *push* it to them. Push upgrades happen automatically, that is, the next time your customer logs in, they have the updates. Let's try that.

1. Log in to your dev org.
2. In the upper right corner, click **Developing Hello World, version 1.0**.

Developing Hello World, version 1.0



3. On the Package Detail page, click **Push Upgrades**.
4. Click **Schedule Push Upgrades**.
5. From the Patch Version drop-down list, select the patch version to push.
6. In the **Scheduled Start Date** field, enter today's date.
7. In the Select Target Organizations section, select your test org.
8. Click **Schedule**.

And you've done it! You pushed a patch release to your subscriber so that they automatically get your updates. You should verify that your customers received the patch to ensure it was installed successfully.

Tell Me More....

Beta versions aren't eligible for push upgrades. You must uninstall a beta and then install a new one.

You can also exclude specific subscriber orgs from the push upgrade by entering the org IDs, separated by a comma, in the Push Upgrade Exclusion List.

Updating Your App Summary

In this tutorial you learned how to update your app in a patch org and push that update to your customers. You started by creating a patch organization that was specific to a released package version. Then you modified your app, uploaded it, and scheduled the push upgrade to your customers.

Congratulations, you're done! Or have you really just begun? You can modify your existing app to be anything you want it to be, or create a new dev org in the Environment Hub and build another app. You can use the same sales and test orgs and everything else you've configured to publish and manage many more apps. You're on your way to ISVforce success!

CHAPTER 3 Grow Your AppExchange Business

In this chapter ...

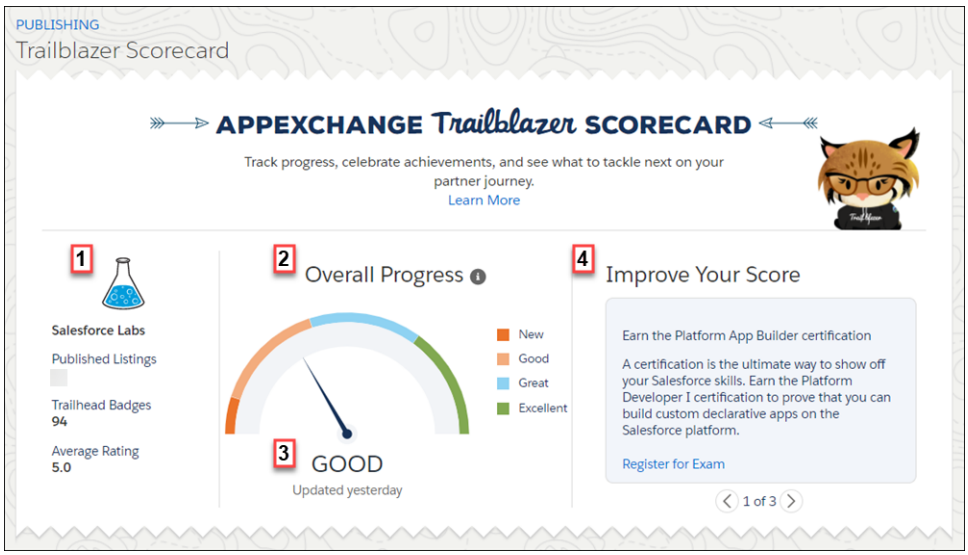
- AppExchange Trailblazer Scorecard
- AppExchange Trailblazer Scorecard Categories
- Create Your AppExchange Trailblazer Scorecard Strategies and Goals
- Get Started with the AppExchange Trailblazer Scorecard
- AppExchange Trailblazer Scorecard FAQ

Grow your AppExchange business faster with the AppExchange Trailblazer Scorecard. The scorecard tracks goals and milestones across the partner journey and recommends steps to take to build a thriving Salesforce business. The scorecard is available in the Publishing Console on the Partner Community.

AppExchange Trailblazer Scorecard

The AppExchange Trailblazer Scorecard is the easy way to track your progress as a Salesforce partner. As you grow your business with us, the scorecard highlights your achievements and recommends tasks that can help you take your partnership to the next level. Learn how to navigate the scorecard, and see how we measure your progress.

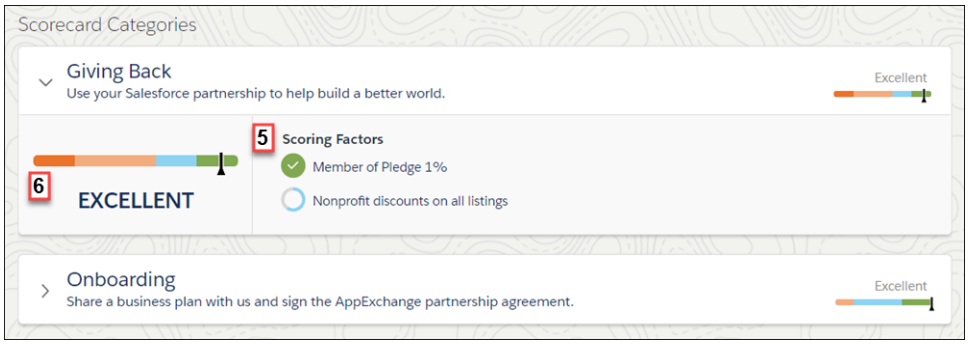
Note: The AppExchange Trailblazer Scorecard is available to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, visit https://partners.salesforce.com.



The top of the scorecard summarizes your company's overall progress and provides recommendations for improving your score. Let's take a closer look at the elements in this section.

Element	Shows You
Company Details (1)	Metrics related to the health of your partnership, such as the average rating of your product.
Overall Progress (2)	Your overall progress based on achievements in categories across the partner journey, such as setting up your AppExchange listing.
Last Updated Date (3)	The last time your scorecard was updated. Scorecards are updated daily.
Recommendations (4)	The top three actions that you can take to improve your score, such as offering discounted pricing for nonprofits on your listing. As you accomplish these tasks, new recommendations are added so that you can keep making progress.

Let's take a deeper look at a scorecard category.



A category contains scoring factors (5) to measure progress within the category (6). Some scoring factors look at one-time milestones along the partner journey. For example, you joined the Pledge 1% program. Others, like total product installs, are evaluated on an ongoing basis. For details about how a scoring factor is measured, hover over the factor.

Progress for scoring factors has four levels: New, Good, Great, and Excellent. For factors that consider one-time milestones, your progress level remains New until you accomplish the milestone. Then your progress level switches to Excellent. For factors that are evaluated on an ongoing basis, there are no fixed rules about moving between levels. For example, we don't publish a fixed number of product installs for achieving the Good progress level. Rather, the guidance is general: the more, the better.

AppExchange Trailblazer Scorecard Categories

To determine overall AppExchange Trailblazer Scorecard progress, we consider your progress in categories that represent important aspects of your Salesforce partnership experience. Learn about the categories and how they contribute to your overall progress.

Category	Considerations	Impact on Overall Progress
Onboarding	Whether you shared a business plan with us and signed a partnership agreement.	Low
Giving Back	Whether you're committed to using your partnership to give back to the community.	Low
Product Security	Whether you submitted a product for security review and passed.	Low
Learning	How many badges and certifications your team has earned.	Medium
Ohana	How often your team shares questions, answers, and ideas with the Salesforce community.	Medium
Listing Setup	Whether your listing has screenshots, demos, and other resources that promote customer engagement.	Medium
Customer Engagement	How often Salesforce customers interact with and provide feedback on your listing.	High

Category	Considerations	Impact on Overall Progress
Installs	How often Salesforce customers install your product and whether installs grow year-over-year. Note: If you've been a partner for less than one year, we don't measure install growth for your product.	High
Revenue	The total annual contract value (ACV) generated by your product and whether ACV grows year-over-year. Note: If you've been a partner for less than one year, we don't measure revenue growth for your product.	High

To explain impact, let's take a closer look at the Onboarding category. This category has two scoring factors: you share a business plan with us, and sign your partnership agreement.

This category is low impact because when you complete these tasks for a listing, you're done. You don't have to revisit the category unless you create another listing. As impact increases, so does your ongoing commitment. For example, we know that it takes time and effort to regularly answer questions on the Trailblazer Community. We want to respect that commitment by increasing the impact of that activity on your overall progress. High-impact categories reflect the most challenging—and rewarding—tasks that you take on as you build your partnership.

Create Your AppExchange Trailblazer Scorecard Strategies and Goals

To accelerate your AppExchange Trailblazer Scorecard progress, combine the scorecard recommendations with a strategy based on your Salesforce partnership experience level.

Experience Level	Number of Published Listings	Strategies
New partner	None	**Go for quick wins** Many scorecard categories focus on one-time tasks, such as signing your partnership agreement or joining the Trailblazer Community. Tackle these tasks before you move on to categories for which scoring factors are evaluated on an ongoing basis. **Connect your badges and certifications**

Experience Level	Number of Published Listings	Strategies
		Many new partners bring years of Salesforce experience with them. If you and your team have already earned Trailhead badges or certifications, connect your accounts to get credit for your hard work.
Veteran partner	One or more	**Check your AppExchange lineup** If your company has multiple listings, all of them are considered when measuring scorecard progress. So don't forget your early adventures. If listings are collecting dust, update them. **Stay active on the Trailblazer Community** The Trailblazer Community is where Salesforce customers go to learn and connect. After you join the community, stop by regularly and share your questions, answers, wisdom, and ideas.

Get Started with the AppExchange Trailblazer Scorecard

View the AppExchange Trailblazer Scorecard to see your company's progress as a Salesforce partner. To allow team members to view the scorecard, assign the Manage Listings permission in the Partner Community. If you have questions about the scorecard, post them on the Partner Community.

View the AppExchange Trailblazer Scorecard

To track the progress of your Salesforce partnership, view your company's AppExchange Trailblazer Scorecard. The scorecard is available in the Publishing Console on the Partner Community.

Grant Access to the AppExchange Trailblazer Scorecard

The Manage Listings permission in the Partner Community controls access to the AppExchange Trailblazer Scorecard. Assign the permission to the people on your team who track your company's performance on AppExchange. If you're the Partner Community admin at your company, you already have access to the scorecard.

Find Answers to AppExchange Trailblazer Scorecard Questions

If you have a question about the AppExchange Trailblazer Scorecard and can't find an answer in the documentation, get help on the Partner Community. We review questions that are posted to the Questions & Answers (Q&A) collaboration group to get you back on track.

View the AppExchange Trailblazer Scorecard

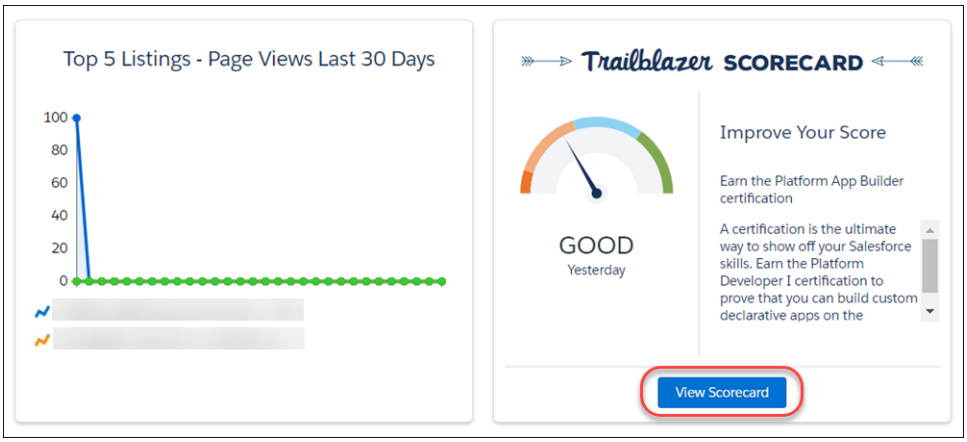
To track the progress of your Salesforce partnership, view your company's AppExchange Trailblazer Scorecard. The scorecard is available in the Publishing Console on the Partner Community.

1. Log in to the Partner Community.
2. Click **Publishing**.
3. Click **Listings**.
4. Click **View Scorecard**.

USER PERMISSIONS

To view the scorecard:

- Manage Listings



To see steps that you can take to improve your score, check the scorecard's recommendations.

Grant Access to the AppExchange Trailblazer Scorecard

The Manage Listings permission in the Partner Community controls access to the AppExchange Trailblazer Scorecard. Assign the permission to the people on your team who track your company's performance on AppExchange. If you're the Partner Community admin at your company, you already have access to the scorecard.

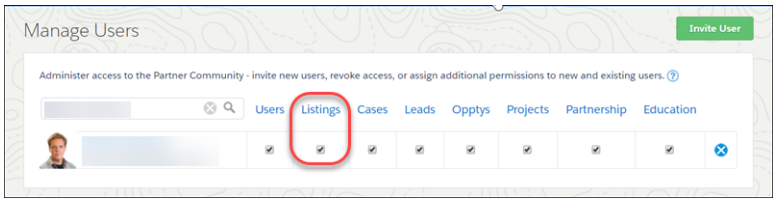
The Manage Listings permission provides full access to the Publishing Console. We suggest assigning the permission to team members whose role also includes managing AppExchange listings.

USER PERMISSIONS

To assign permissions to Partner Community users:

- Manage Users

1. Log in to the Partner Community.
2. Click **Manage Users**.
3. Search for a user at your company.
4. To enable the permission, select the checkbox under Listings.



Find Answers to AppExchange Trailblazer Scorecard Questions

If you have a question about the AppExchange Trailblazer Scorecard and can't find an answer in the documentation, get help on the Partner Community. We review questions that are posted to the Questions & Answers (Q&A) collaboration group to get you back on track.

1. Log in to the Partner Community.
2. Go the Questions & Answers (Q&A) collaboration group.

3. Post your question to the community.

   Tip: Before you post, try searching the group feed using keywords from your question. If someone posted the question already, an answer might be available immediately.

AppExchange Trailblazer Scorecard FAQ

Here are some answers to frequently asked questions about the AppExchange Trailblazer Scorecard.

How often is the AppExchange Trailblazer Scorecard updated?
Why doesn't the AppExchange Trailblazer Scorecard show install or revenue growth for my listing?
Why doesn't the AppExchange Trailblazer Scorecard show installs for my listing?
Why doesn't the AppExchange Trailblazer Scorecard show my Trailhead badges and Salesforce certifications?
Can I use the AppExchange Trailblazer Scorecard if I'm a consulting partner?

How often is the AppExchange Trailblazer Scorecard updated?

The scorecard is updated daily. To see the last time that your scorecard was updated, go to the Overall Progress area.

Why doesn't the AppExchange Trailblazer Scorecard show install or revenue growth for my listing?

We measure install and revenue growth year-over-year. If you joined the Partner Program less than one year ago, or didn't have installs or revenue in the previous year, the scorecard doesn't show progress for these factors.

Why doesn't the AppExchange Trailblazer Scorecard show installs for my listing?

We measure installs only for listings with managed packages. If you distribute your product using another method, such as an unmanaged package, the scorecard doesn't show progress. If you distribute your product using a managed package and you don't see progress, log a case in the Partner Community.

Why doesn't the AppExchange Trailblazer Scorecard show my Trailhead badges and Salesforce certifications?

The most common reason is that a Trailhead or certification account isn't connected to the Partner Community. To connect an account, go to your Partner Community profile. If you've connected an account and you don't see progress, log a case in the Partner Community.

Can I use the AppExchange Trailblazer Scorecard if I'm a consulting partner?

If you're a consulting partner with a product listing, such as an app listing, you can use the scorecard to see how the listing is performing. We still use your Partnership Value Score (PVS) to determine your consulting program tier and benefits. For more information, go to the Partner Community.

CHAPTER 4 Design and Build Your Solution

In this chapter ...

- Overview of Packages
- Components Available in Managed Packages
- About API and Dynamic Apex Access in Packages
- Architectural Considerations for Group and Professional Editions
- Connected Apps
- Environment Hub
- Developer Hub
- Notifications for Package Errors

Discover the architectural concepts that influence AppExchange solution design. Learn how to plan, build, and package your solution for customers.

Overview of 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

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.

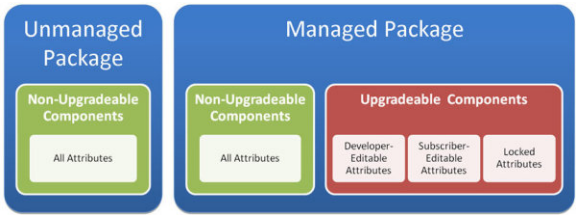
Managed packages also offer the following benefits:

- 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

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.

The following definitions illustrate these concepts:

Unmanaged and Managed 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.

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 on page 275.
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. This type of release is considered a patch release on page 275.
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. For details, see Delete Components in Managed Packages on page 281.

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 62. 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.
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**.

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

Developing and Distributing Unmanaged Packages

Unmanaged packages are traditionally used for distributing open-source projects to developers, or as a one time drop of applications that require customization after installation. You should never use unmanaged packages for sandbox to production migration. Instead, use theForce.com IDE or the Ant Migration Tool. If you're using Enterprise, Unlimited, or Performance Edition, see Change Sets.

SEE ALSO:

Components Available in Unmanaged Packages

Create and Upload an Unmanaged Package

Use the following procedure to upload an unmanaged package through the UI. (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*.)

1. Create the package:
   a. From Setup, enter `Packages` in the `Quick Find` box, then select **Packages**.
   b. Click **New**.
   c. Fill in the details of the package.
   d. Click **Save**.
2. Add the necessary components for your app.
   a. Click **Add Components**.
   b. From the drop-down list, choose the type of component.
   c. Select the components you want to add.

      Note: Some components cannot be added to Managed - Released packages. For a list of packageable components, see Components Available in Managed Packages on page 29. If you add S-Controls and documents, keep in mind that their combined size must be less than 10 MB. Also, S-controls cannot be added to packages with restricted API access.

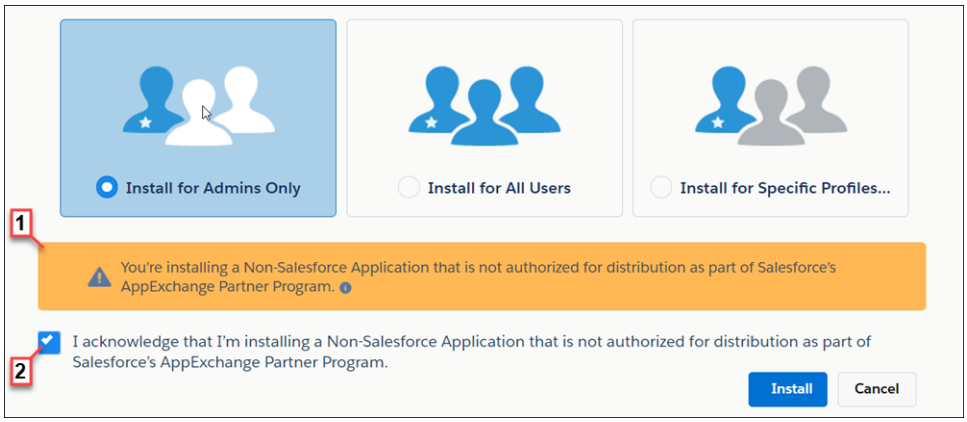
   d. Click **Add To Package**.
   e. Repeat these steps until you have added all the components you want in your package.

   Note: Some related components are automatically included in the package even though they might not 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, see Components Automatically Added to Packages on page 44.
3. Click **Upload**.

You will receive an email that includes an installation link when your package has been uploaded successfully. Wait a few moments before clicking the installation link or distributing it to others, as it might take a few minutes for it to become active.

Install Notifications for Unauthorized Managed Packages

When you distribute a managed package that the AppExchange Partner Program hasn't authorized, we notify customers during the installation process. The notification is removed after the package is approved.



The notification appears when customers configure the package installation settings (1). Before customers install the package, they must confirm that they understand that the package isn't authorized for distribution (2).

The notification displays when a managed package:

- Has never been through security review or is under review
- Didn't pass the security review
- Isn't authorized by the AppExchange Partner Program for another reason

If the AppExchange Partner Program approves the package, it's authorized for distribution, and the notification is removed. When you publish a new version of the package, it's automatically authorized for distribution.

For information about the AppExchange Partner Program and its requirements, visit the Salesforce Partner Community.

Components Available in Managed Packages

A component is a part of a managed 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 managed packages, you can upgrade some components.

Not all components can be packaged for distribution. If you create an app that uses components that aren't packageable, your subscribers have to create and configure the components after they install your app. If ease of installation is an important concern for your subscribers, keep the 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 is not upgraded.
- **Yes**: The component is upgraded.

Subscriber Deletable

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.

Developer Deletable

A developer can delete some components after the package is uploaded as Managed - Released. Deleted components are not deleted in the subscriber's org during a package upgrade. The Protectable attribute contains more details on deleting components.

- **No**: The developer cannot 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 the subscriber's org.

- **No**: The component cannot 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. The information in the components that you package and publish could be visible to users on 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	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
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

Component	Upgradeable	Subscriber Deletable	Developer Deletable	Protectable	IP Protection
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 Label	Yes	No	Yes, if protected	Yes	No
Custom Metadata Records	Yes	No	Yes	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]	No	Yes
Dashboard	No	Yes	Yes	No	No
Document[7] (10 MB limit)	No	Yes	Yes	No	No
Email Template	No	Yes	Yes	No	No
External Data Source	Yes	No	No	No	No

[1] Requires a Service Cloud license.

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
Lightning page	Yes	No	No	No	No
Flow	Yes	Yes	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
Letterhead	No	Yes	Yes	No	No
Lightning Application	Yes	No	No	No	No
Lightning Component	Yes	No	No	No	No
Lightning Event	Yes	No	No	No	No
Lightning Interface	Yes	No	No	No	No
Lightning Web Component	Yes	No	No	No	No
List View	No	Yes	Yes	No	No
Named Credential[5]	Yes	No	No	No	No
Page Layout	No	Yes	Yes	No	No
Path Assistant	Yes	No	No	No	No
Permission Set	Yes	Yes[2]	Yes[3]	No	No
Platform Cache	No	No	No	No	No
Process	See Flow				
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

Component	Upgradeable	Subscriber Deletable	Developer Deletable	Protectable	IP Protection
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
Snap-in Deployment[6][2]	No	No	No	No	No
Tab	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 with a new version. 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 cannot 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 part of 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. Alternatively, a subscriber can create a named credential in the subscriber org. However, the subscriber must use the same name as the named credential specified in the External Services registration that references it. Named credentials can be created manually or with Apex. Ensure that the named credential is added to a package so that subscriber orgs can install it. Upon installation, the subscriber org can use the Apex callouts generated by the External Services registration process.

[2] Requires a Service Cloud license.

[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.

Component Attributes and Behaviors

Only some attributes of a component are upgradeable. Many components also behave differently or include additional restrictions in a managed package. It's important to 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.

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

SEE ALSO:

Editing Components and Attributes After Installation

Components Automatically Added to Packages

Delete Components in Managed Packages

Components Available in Unmanaged Packages

Not all components can be packaged for distribution. The following table lists the components that are available in an unmanaged package, how the component is included in the package, and whether the component supports automatic renaming.

Packaged Explicitly or Implicitly

Components can be added either explicitly or implicitly. Explicit components must be included directly in the package, while implicit components are automatically added. For example, if you create a custom field on a standard object, you must explicitly add the custom field to your package. However, if you create a custom object and add a custom field to it, the field is implicitly added to the package when you add the custom object.

- **Explicitly**: The component must be manually added to the package.
- **Implicitly**: The component is automatically added to the package when another dependent component, usually a custom object, is added.

Automatic Renaming

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.

Component	Packaged Explicitly or Implicitly	Automatic Renaming
Reporting Snapshot	Explicitly	Yes
Apex Class	Explicitly	No
Apex Sharing Reason	Implicitly On an extension: Explicitly	No
Apex Sharing Recalculation	Implicitly	No

Component	Packaged Explicitly or Implicitly	Automatic Renaming
Apex Trigger	On a standard or extension object: Explicitly On an object in the package: Implicitly	No
Application	Explicitly	No
Custom Button or Link	On a standard object: Explicitly On a custom object: Implicitly	No
Custom Field	On a standard object: Explicitly On a custom object: Implicitly	No
Custom Label	Implicitly	No
Custom Object	Explicitly	No
Custom Permission	Implicitly With required custom permissions: Explicitly	No
Custom Report Type	Explicitly	No
Custom Setting	Explicitly	No
Dashboard	Explicitly In a folder: Implicitly	Yes
Document* (10 MB limit)	Explicitly In a folder: Implicitly	Yes
Email Template	Explicitly In a folder: Implicitly	Yes
External Data Source	Explicitly Referenced by an external object: Implicitly Assigned by a permission set: Implicitly	No
Flow Definition	Implicitly	No
Folder	Explicitly	Yes
Home Page Component	Explicitly	No
Home Page Layout	Explicitly	No
Letterhead	Explicitly	Yes
Lightning Application	Explicitly	No
Lightning Component	Explicitly	No
Lightning Event	Explicitly	No

Component	Packaged Explicitly or Implicitly	Automatic Renaming
Lightning Interface	Explicitly	No
List View	On a standard object: Explicitly On a custom object: Implicitly	Yes
Named Credential	Explicitly	No
Page Layout	On a standard object: Explicitly On a custom object: Implicitly	No
Record Type	On a standard object: Explicitly On a custom object: Implicitly	No
Report	Explicitly In a folder: Implicitly	Yes
S-Control* (10 MB limit)	Explicitly	No
Static Resource	Explicitly	No
Tab	Explicitly	No
Translation	Explicitly	No
Validation Rule	On a standard object: Explicitly On a custom object: Implicitly	No
Visualforce Component	Explicitly	No
Visualforce Page	Explicitly	No
Workflow Email Alert	Explicitly	No
Workflow Field Update	Explicitly	No
Workflow Outbound Message	Explicitly	No
Workflow Rule	Explicitly	No
Workflow Task	Explicitly	No

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

SEE ALSO:

Components Automatically Added to Packages

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 • Code		• Name
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	• Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
		• Grant Access Using Hierarchy • Search Layouts	
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 • Locations • Logo Image URL • OAuth Scopes	• ACS URL • Entity ID • IP Relaxation • Manage Permission Sets • Manage Profiles • Mobile Start URL • Permitted Users • Refresh Token Policy • SAML Attributes • Service Provider Certificate • Start URL • Subject Type	• API Name • Created Date/By • Consumer Key • Consumer Secret • Installed By • Installed Date • Last Modified Date/By • Version
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	• 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

Component	Developer Editable	Subscriber and Developer Editable	Locked
	• Related List Label • Required • Roll-Up Summary Filter Criteria		
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 • Enhanced Lookup • Grant Access Using Hierarchy • Search Layouts • Track Field History	• Object Name • Record Name Data Type • Record Name Display Format
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

Component	Developer Editable	Subscriber and Developer Editable	Locked
Document		• All attributes except Document Unique Name	• Document Unique Name
Email Template		• All attributes except Email Template Name	• Email Template Name
External Data Source	• Type	• Auth Provider • Certificate • Custom Configuration • Endpoint • Identity Type • OAuth Scope • Password • Protocol • Username	• Name
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
Letterhead		• All attributes except Letterhead Name	• Letterhead Name
Lightning Application	• API Version • Description • Label • Markup		Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
Lightning Component	• API Version • Description • Label • Markup		Name
Lightning Event	• API Version • Description • Label • Markup		Name
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	• Allow Merge Fields in HTTP Body • Allow Merge Fields in HTTP Header • Auth Provider • Certificate • Generate Authorization Header • Identity Type • OAuth Scope • Password • Protocol • Username	• Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
Page Layout		• All attributes except Page Layout Name	• Page Layout Name
Permission Set	• Description • Label • Custom object permissions • Custom field permissions • Apex class access settings • Visualforce page access settings		• Name
Platform Cache			• All attributes
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	• Tab Style	• Tab Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
	• Splash Page Custom Link • Type • URL • 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 • 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

Component	Developer Editable	Subscriber and Developer Editable	Locked
Workflow Task		• Assign To • Comments • Due Date • Priority • Record Type • Status	• Subject

Components Automatically Added to Packages

When adding components to your package, some related components are automatically added, if required. 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 might be automatically included, review the following list:

When you add this component:	These types of components might be automatically included:
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, as well as anything else that is directly referenced by the Apex class Note: 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.
Apex trigger	Custom fields, custom objects, and any explicitly referenced Apex classes, as well as anything else that is directly referenced by the Apex trigger
Article type	Custom fields, the default page layout
Compact layout	Custom fields
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

When you add this component:	These types of components might be automatically included:
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	Folder, letterhead, custom fields, and documents (stored as images on the letterhead or template)
Field set	Any referenced fields
Lightning page	Any associated actions
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
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.

When you add this component:	These types of components might be automatically included:
Page layout	Actions, custom buttons, custom links, s-controls, and Visualforce pages
Permission set	Any custom permissions, external data sources, Visualforce pages, 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, might not display in the list of package components, but are included and install with the other components.

Special Behavior of Components in Packages

When you're building an app for distribution, it's important to consider how packaging affects your app and its components. Use the following information 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.
- For more information on the attributes of each component in packages, see the component attributes table.
- Component names must be unique within an org. To ensure that your component names don't conflict with those in an installer's org, use a managed package so 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 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, 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 cannot 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 cannot 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 will be 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

- When explicitly referencing a picklist value in code, keep in mind that subscribers can add, edit, and delete picklist field values for custom fields. Carefully consider this possibility when explicitly referencing a picklist value in code. Developers can add and delete picklist values, but new picklist values are not installed into the subscriber's org for existing fields during a package upgrade. Picklist values deleted by the developer are still available in the subscriber's org.
- Package upgrades remove dependent picklist values that are saved in a managed custom field.
- 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 cannot be added after the object is uploaded in a Managed - Released package.

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 org 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 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, you receive an error message.

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 does not need 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 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 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 does not 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 does not 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 does not support divisions.

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 re-enter 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.

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 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 is not 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 is not 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 notes 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.

- If you're translating a flow from a managed package, the flow's Master Definition Name won'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, 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 org 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 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

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 org 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 org 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 org is not 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 is not 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.

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

- Subscribers can rename or delete picklist field values. Keep this in mind 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.
- Global value sets can be added to developer and subscriber orgs. Global value sets have the following behavior during a package upgrade:
  - Labels for the values in a global value set change in the subscriber orgs to match the values in the upgrade package.
  - New values in the upgrade package's global value set are added to the subscriber orgs.
  - The package upgrade overrides the Active and Inactive value settings in the subscriber orgs. If a subscriber set a value to Inactive, and the upgrade package includes the same value as Active, the upgrade changes the setting in the subscriber org to Active.
  - The package upgrade does not override the default values in the subscriber orgs.

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 object permissions
- External object permissions
- Custom field permissions
- Custom 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 is 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 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 org is not using record types on the object or does not 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 does not support record types or the object does not 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 cannot 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 have been moved to a private folder or to the Unfiled Public Reports folder.
- When a package is installed into an org that does not 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 additional 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 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 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 are 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 master 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 are not supported for general-purpose distribution. While Analytics unmanaged packages work as expected within the constraints of Salesforce unmanaged packages, they aren't subject to the same level of 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.

- Analytics Admin permissions are required to create a package but not for deployment, which requires only Salesforce admin permissions.
- There is no spidering between datasets and dataflows, meaning there is no dependency following. When packaging both, they must be added manually. If they are not, an error appears during deployment. The same is true for change sets—when packaging both datasets and dataflows, add them manually.
- Because views are user-specific, they aren't included when you package the dashboard.
- 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 are not 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 the Flows action in Process Builder instead.

- Developers can protect some workflow actions. For more information on protected components, see Protected Components.
- 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 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, 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, 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 permissions
- 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 the Flows action in Process Builder instead.

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.

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:

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

- 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.

Metadata Access in Apex Code

Use the `Metadata` namespace in Apex to access metadata in your package.

Your package may need to retrieve or modify metadata during installation or update. The `Metadata` namespace in Apex provides classes that represent metadata types, as well as classes that let you retrieve and deploy metadata components to the subscriber org. These considerations apply to metadata in Apex:

- You can create, retrieve, and update metadata components in Apex code, but you can't delete components.
- You can currently access records of custom metadata types and page layouts in Apex.
- Managed packages not approved by Salesforce can't access metadata in the subscriber org, unless the subscriber org enables the **Allow metadata deploy by Apex from non-certified Apex package version** org preference. Use this org preference when doing test or beta releases of your managed packages.

If your package accesses metadata during installation or update, or contains a custom setup interface that accesses metadata, you must notify the user. For installs that access metadata, notify the user in the description of your package. The notice should let customers know that your package has the ability to modify the subscriber org's metadata.

You can write your own notice, or use this sample:

```
This package can access and change metadata outside its namespace in the Salesforce
org where it's installed.
```

Salesforce verifies the notice during the security review.

For more information, see Metadata in the *Apex Developer Guide*.

About Permission Sets and Profile Settings

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 permissions • Custom tab visibility settings • Apex class access • Visualforce page access • External data source access 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.	• Assigned custom apps • Assigned connected apps • Tab settings • Page layout assignments • Record type assignments • Custom object permissions • External object permissions • Custom field permissions • Custom permissions • Apex class access • Visualforce page access • 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	No.

Behavior	Permission Sets	Profile Settings
	master-detail relationship where the master is a standard object.	
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 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.	None. In a subscriber org, the installation overrides the profile settings, not their user licenses.
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.

Custom Profile Settings

When building your AppExchange app, create profiles to define how users access objects and data, and what they can do within your app. For example, profiles specify custom object permissions and the tab visibility for your app. When installing or upgrading your app, admins can associate your custom profiles with existing non-standard profiles. Permissions in your custom profile that are related to new components created as part of the install or upgrade are added to the existing profile. The security settings associated with standard objects and existing custom objects in an installer's organization are unaffected.

Consider these tips when creating custom profiles for apps you want to publish.

- Give each custom profile a name that identifies the profile as belonging to the app. For example, if you're creating a Human Resources app named "HR2GO," a good profile name would be "HR2GO Approving Manager."
- If your custom profiles have a hierarchy, use a name that indicates the profile's location in the hierarchy. For example, name a senior-level manager's profile "HR2GO Level 2 Approving Manager."
- Avoid custom profile names that can be interpreted differently in other organizations. For example, the profile name "HR2GO Level 2 Approving Manager" is open to less interpretation than "Sr. Manager."
- Provide a meaningful description for each profile. The description displays to the user installing your app.

Alternatively, you can use permission sets to maintain control of permission settings through the upgrade process. Permission sets contain a subset of profile access settings, including object permissions, field permissions, Apex class access, and Visualforce page access. These permissions are the same as those available on profiles. You can add a permission set as a component in a package.

Note: In packages, assigned apps and tab settings aren't included in permission set components.

Protecting Your Intellectual Property

The details of your custom objects, custom links, reports, and other installed items are revealed to installers so that they can check for malicious content. However, revealing an app's components prevents developers from protecting some intellectual property.

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 your components are available on AppExchange, you cannot recall them from anyone who has installed them.
- The information in the components that you package and publish might be visible to customers. Use caution when adding your code to a formula, Visualforce page, or other component that you cannot hide in your app.
- 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.
- 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.

Creating Packaged Applications with Chatter

The objects, field settings, and field settings history of Chatter are packageable. However, an object's field is only tracked if the object itself is tracked. For example, you can create a new custom field on the Account standard object, but the field will only be tracked if you have enabled feed tracking on Accounts.

When developing applications that use Chatter, it's important to be aware that some organizations might not have Chatter enabled. By default, when you upload Chatter applications, the package is only available to organizations that have Chatter enabled. You can change this behavior and allow organizations to install the package even if they don't have Chatter. Note the following:

- You must use a managed package. Unmanaged packages that include Chatter functionality can only be installed in organizations that have Chatter enabled.
- DML operations and SOSL, and SOQL calls will throw a runtime exception if the subscriber organization does not have Chatter enabled. You must catch and handle any Apex exceptions that are thrown as a result of the missing Chatter feature. These exceptions are of the type `REQUIRED_FEATURE_MISSING_EXCEPTION` for SOSL and SOQL calls. For DML calls, you must check for the specific `REQUIRED_FEATURE_MISSING` status code on a DML Exception.

- When you upload the package, deselect the Chatter required checkbox (this is automatically selected if you have an Apex reference to Chatter).

  Note: If the Chatter required checkbox can't be deselected, then some component in the package has a special requirement for Chatter. This can happen, for example, if you package a custom report type that relies on Chatter. If the Chatter-required checkbox can't be disabled, then the package can only be installed in organizations that have Chatter enabled.

The following example tries to post to feeds and get a user's feed. If Chatter is not enabled in the organization, the code catches the `REQUIRED_FEATURE_MISSING` exception. Note that this is an incomplete code example and does not run.

```
public void addFeedItem(String post, Id objId) {
   FeedItem fpost = new FeedItem();
   // Get the parent ID of the feed
   fpost.ParentId = objId;
   fpost.Body = post;
   try{
      insert fpost;
   } catch (System.DmlException e) {
         for (Integer i = 0; i < e.getNumDml(); i++) {
         // Chatter not endabled, do not insert record
            System.assertEquals(StatusCode.REQUIRED_FEATURE_MISSING, e.getDmlType(i));
           System.Debug('Chatter not enabled in this organization:' + e.getDMLMessage());

        }
   }
}

  public List<NewsFeed> getMyFeed() {
    List<NewsFeed> myfeed;
     try{
         myfeed = [SELECT Id, Type, CreatedById, CreatedBy.FirstName,CreatedBy.LastName,
                   CreatedDate, ParentId, Parent.Name,FeedItemId, Body,
                   Title, CreatedById, LinkUrl,
                    (SELECT Id, FieldName, OldValue, NewValue
                     FROM FeedTrackedChanges ORDER BY Id DESC),
                       (SELECT Id, CommentBody, CreatedDate, CreatedById,
                        CreatedBy.FirstName, CreatedBy.LastName
                        FROM FeedComments ORDER BY CreatedDate DESC, ID DESC LIMIT 10)
                   FROM NewsFeed
                   ORDER BY CreatedDate DESC, ID DESC LIMIT 20];
     } catch(System.RequiredFeatureMissingException e){
        // The above has returned an empty NewsFeed
        // Chatter is not enabled in this organization
        myfeed = new List<NewsFeed>{};
              System.Debug('Chatter not enabled in organization:' + e.getMessage());
     }
     return myfeed;
   }
```

Matching the Salesforce Look and Feel

Apps that resemble the Salesforce user interface look and feel are instantly more familiar to users and easy to use. The easiest way to model the design of your app after the Salesforce user interface look and feel is to use Visualforce. When you use a standard controller

with a Visualforce page, your new page takes on the style of the associated object's standard tab in Salesforce. For more information, see Using Salesforce Styles in the *Visualforce Developer's Guide*.

Maintaining My Domain URLs

Build packages that support both instance and instanceless My Domain URLs. As a best practice, replace hard-coded references that contain the instance name in the URL with relative URLs, which are non-instance specific.

A critical update in Spring '18 removed instance names from My Domain URLs for Visualforce, Community Builder, Site.com Studio, and content files. Update existing packages to support instanceless URLs before the critical update's auto-activation date on March 16, 2019. Otherwise, users will have page functionality issues when using outdated packages.

This URL is a hard-coded reference: `<MyDomain>--<PackageName>.<instance>.visual.force.com/{Case.Id}`

When the critical update is enabled, the URL becomes `<MyDomain>--<PackageName>.visualforce.com/{Case.Id}`

We recommend using relative URLs to avoid any issues with domain names. To make this hard-coded reference a relative URL, remove "`<MyDomain>--<PackageName>.<instance>.visual.force.com`" so that the URL becomes `/{Case.Id}`

For more information, see My Domain URL Changes in the *Identity Implementation Guide*.

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.

About API and Dynamic Apex Access in Packages

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.

EDITIONS

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

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

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.

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 63.

   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 63.

   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.

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

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

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.

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 45.0, 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 the Partner WSDL

The Partner Web Services WSDL is used for client applications that are metadata-driven and dynamic in nature. It is particularly—but not exclusively—useful to Salesforce partners who are building client applications for multiple organizations. As a loosely typed representation of the Salesforce data model that works with name-value pairs of field names and values instead of specific data types, it can be used to access data within any organization. This WSDL is most appropriate for developers of clients that can issue a query call to get information about an object before the client acts on the object. The partner WSDL document needs to be downloaded and consumed only once per version of the API.

For more information about the Partner WSDL, see Using the Partner WSDL in the *SOAP API Developer Guide*.

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.

EDITIONS

Available in: Salesforce Classic

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

USER PERMISSIONS

To download a WSDL:
- Customize Application

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

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 45.0, 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.

Work with Services Outside of Salesforce

You might want to update your Salesforce data when changes occur in another service. Likewise, you might also want to update the data in a service outside of Salesforce based on changes to your Salesforce data. For example, you might want to send a mass email to more contacts and leads than Salesforce allows. You can use an external mail service that allows users to build a recipient list of names and email addresses using the contact and lead information in your Salesforce organization.

An app built on the Lightning Platform platform can connect with a service outside of Salesforce in many ways. For example, you can:

- create a custom link or custom formula field that passes information to an external service.

- use the Lightning Platform API to transfer data in and out of Salesforce.
- use an Apex class that contains a Web service method.

Before any Visualforce page, Apex callout, or JavaScript code using XmlHttpRequest in an s-control or custom button can call an external site, that site must be registered in the Remote Site Settings page, or the call fails. For information on registering components, see Configure Remote Site Settings.

Warning: Do not store usernames and passwords within any external service.

Provisioning a Service External to Salesforce

If your app links to an external service, users who install the app must be signed up to use the service. Provide access in one of two ways:

- Access by all active users in an organization with no real need to identify an individual
- Access on a per user basis where identification of the individual is important

The Salesforce service provides two globally unique IDs to support these options. The user ID identifies an individual and is unique across all organizations. User IDs are never reused. Likewise, the organization ID uniquely identifies the organization.

Avoid using email addresses, company names, and Salesforce usernames when providing access to an external service. Usernames can change over time and email addresses and company names can be duplicated.

If you are providing access to an external service, we recommend the following:

- Use Single Sign-On (SSO) techniques to identify new users when they use your service.
- For each point of entry to your app, such as a custom link or web tab, include the user ID in the parameter string. Have your service examine the user ID to verify that the user ID belongs to a known user. Include a session ID in the parameter string so that your service can read back through the Lightning Platform API and validate that this user has an active session and is authenticated.
- Offer the external service for any known users. For new users, display an alternative page to collect the required information.
- Do not store passwords for individual users. Besides the obvious security risks, many organizations reset passwords on a regular basis, which requires the user to update the password on your system as well. We recommend designing your external service to use the user ID and session ID to authenticate and identify users.
- If your application requires asynchronous updates after a user session has expired, dedicate a distinct administrator user license for this.

Architectural Considerations for Group and Professional Editions

Salesforce CRM is offered in five tiers, or editions:

- Group Edition (GE)*
- Professional Edition (PE)
- Enterprise Edition (EE)
- Unlimited Edition (UE)
- Performance Edition (PXE)*

Note: Group and Performance Editions are no longer sold. For a comparison chart of editions and their features, see the Salesforce Pricing and Editions page.

If you plan to sell your app to existing Salesforce customers, it's important to understand the differences between these editions because they will affect the design of your app. It's convenient to think about them in clusters, GE/PE and EE/UE/PXE, as the editions in each cluster have similar functionality. For example, you might only want to support EE/UE/PXE if your app requires certain objects and features

that aren't available in GE/PE. Also, instead of a single solution that supports all editions, you can have a tiered offering. This would consist of a basic solution for GE/PE and an advanced one for EE/UE/PXE customers that takes advantage of the additional features.

EE/UE/PXE have the most robust functionality. They support Lightning Platform platform licenses in addition to Salesforce CRM licenses. If your app doesn't require Salesforce CRM features (such as Leads, Opportunities, Cases, etc.), Lightning Platform platform licenses provide you with the most flexibility in deploying your app to users who might not normally be Salesforce users. Your app is still subject to the edition limits and packaging rules.

GE/PE don't contain all of the functionality that you can build in a Developer Edition (DE). Therefore, an application developed in your DE organization might not install in a GE/PE organization. If you're designing an application to work specifically in GE/PE, you must be aware of how these editions differ.

There are a number of other considerations to keep in mind when deciding whether to support these editions. Lightning Platform platform licenses cannot be provisioned in GE/PE organizations. This means that only existing Salesforce CRM users can use your app. There are some features that aren't available in GE/PE. There are several special permissions available to eligible partner apps that overcome these limitations.

See the following sections for available features, limits, and other design considerations.

- Features in Group and Professional Editions
- Limits for Group and Professional Editions
- Access Control in Group and Professional Editions
- Using Apex in Group and Professional Editions
- API Access in Group and Professional Editions
- Designing Your App to Support Multiple Editions
- Sample Design Scenarios

Features in Group and Professional Editions

The easiest way to determine which features and objects are available in a particular edition is by reviewing the Edition Comparison Table. You can also look up which editions support a specific feature or object by searching the online help. It's important that you check these resources before you start designing your app to make an informed decision on which editions to target. When you're finished building your app, we recommend that you test it by installing your package in GE and PE test orgs to ensure that everything functions properly.

The following table shows the key differences between GE and PE.

Feature	Group Edition	Professional Edition
Assets	No	Yes
Campaigns	No	Yes
Contracts	No	Yes (with the Sales Cloud)
Forecasts	No	Yes (no Opportunity Splits or Custom Field forecasts)
Ideas	No	Yes
Products	No	Yes
Solutions	No	Yes
Record types	No	Yes

Feature	Group Edition	Professional Edition
Permission sets	Yes	Yes
Custom profiles	No	Yes
Custom report types	No	Yes
Workflow and approvals	No	No (See note.)
Apex code	See note.	See note.
Sharing rules	No	Yes (for some features)
API	See note.	See note.
Sites	No	No

Note:

- All listed features are available in DE.
- As a partner, workflows within your application run in a Professional Edition org. However, customers can't create their own workflows. They must purchase the feature directly from Salesforce.
- A client ID allows your app to use the API for integration to composite apps. For more information, see Using Apex in Group and Professional Editions and API Access in Group and Professional Editions.

Limits for Group and Professional Editions

All Salesforce editions have limits that restrict the number of apps, objects, and tabs that can be used. For details on the limits for various editions, see the Edition Limits Table.

For partners who are enrolled in the ISV Program, any managed package publicly posted on the AppExchange no longer counts against the apps/objects/tabs limits for your Salesforce Edition. This effectively means that ISV partners no longer have to worry about package installation failures because of apps/objects/tabs limits being exceeded. This feature is automatically enabled after your app passes the security review.

Access Control in Group and Professional Editions

Group Edition doesn't support field-level security or custom profiles. You can manage field-level security by using the page layout for each object instead. When customers install your app, they can't define which profiles have access to what. Ensure that your design works for the Standard User Profile. Permission sets can be installed but not updated in Group and Professional Edition orgs.

Because the page layout handles field level security, add any fields you want to be visible to the page layout. For fields to be accessible via the API or Visualforce, add them to the page layout.

Using Apex in Group and Professional Editions

Your app can contain business logic such as classes, triggers, email services, etc. written in Apex. As a general rule, Apex is not supported in GE/PE, so it will not run in these editions. However, Apex developed as part of an ISV app and included in a managed package can run in GE/PE, even though those editions do not support Apex by default.

You must be an eligible partner with salesforce.com and your app has to pass the security review. The appropriate permissions will automatically be enabled after you pass the security review.

Here are some important considerations for using Apex in GE/PE.

- GE/PE customers can't create or modify Apex in your app; they can only run the existing Apex.
- Your Apex code should not depend on features and functionality that exist only in DE, EE, UE, or PXE, or your app will fail to install.
- Make sure to use REST if you plan to expose an Apex method as a Web service. Apex classes that have been exposed as a SOAP Web service can't be invoked from an external web app in GE/PE.
- Using Apex to make Web service callouts is allowed in GE/PE. For instance, if you're planning to make a Web service callout to an external Web service, as long as the managed package is authorized, these classes will function in GE/PE.

API Access in Group and Professional Editions

API access is not normally supported in GE and PE orgs. However, after your app passes the security review, you're eligible to use some APIs for building composite applications.

- Currently, the standard Data SOAP and REST APIs are supported for GE and PE apps, and Metadata API is supported in PE apps. To request API access, see How do I get an API token for my app? You can also contact Salesforce to request that a connected app be whitelisted to use the REST API in GE or PE orgs.
- Other APIs, such as the Bulk API and Apex methods exposed as SOAP Web services, remain unavailable.
- You can enable REST-based Web services using connected app consumer whitelisting.
- You can enable SOAP-based Web services, including Metadata API, using an API token called a Client ID, which is appended to your SOAP headers in integration calls. This special key enables your app to make calls to GE and PE orgs for Data API and PE orgs for Metadata API, even if the customer does not have API access.

The Client ID has these properties.

- You can't use the Client ID with the AJAX Toolkit in custom JavaScript, S-controls, or anywhere in your app where its value would be exposed to the end customer.
- For development purposes, GE and PE orgs created via the Environment Hub already have the Metadata API and SOAP API (Data API) enabled. You can then develop and test your app before the security review. After your app passes the security review and you obtain an API token, test your app again to ensure that it's working correctly.
- The Client ID grants GE and PE access to the SOAP API, and PE access to the Metadata API. With the Metadata API, you can dynamically create various components that you typically create in Setup. For instance, you can create a custom field dynamically in a PE organization with the API token.

This table shows which APIs are accessible when using GE and PE and the method of access.

API	Access to GE and PE
Web Services (SOAP)	Yes, with token
Apex methods exposed as Web services (SOAP)	No
Web services (REST)	Yes, with connected app consumer whitelisting
Apex methods exposed as Web services (REST)	Yes, with connected app consumer whitelisting
Chatter REST API	Yes
Metadata API	Yes, with token
Bulk API	No

API	Access to GE and PE
Data Loader tool (uses SOAP Web services)	No, can't set the token

Accessing the REST API in Group and Professional Editions

The Lightning Platform REST API provides you a powerful, convenient, and simple API for interacting with Lightning Platform. Qualified partners can request salesforce.com to enable your application for REST API calls to GE/PE organizations. To get access to the REST API, you must meet these conditions.

- Access to the Partner Community – If you're new, please learn about and join one of the ISV Partner Programs.
- Pass the security review – All applications enrolled in the AppExchange and/or OEM Program must go through a periodic security review.
- Access to Salesforce Developer Edition – If you don't already have access to a DE organization, you can get the Partner Developer Edition from the Environment Hub.

To request the REST API token:

1. Create a new connected app from your DE organization. Log in to salesforce.com with your developer account. From Setup, enter `Apps` in the `Quick Find` box, then select **Apps**, and click **New** in the Connected Apps section.

   Note: We strongly recommend that you do this in an organization you will continue using for a long time, such as the one where you build your managed package or your Trialforce management organization (TMO).

2. Enter the information requested and click **Save**. Saving your app gives you the Consumer Key and Consumer Secret the app uses to communicate with Salesforce.
3. Submit a case from the Partner Community and provide your DE Org ID and the credentials for your connected app.

We'll evaluate your request and enable the appropriate permission. Once this is done, you'll receive a case notification from us. Please wait 24 hours to make sure the permission is completely activated. Your client_id (or Consumer Key) and client_secret (or Consumer Secret) will be checked against the information you submit via the case during the OAuth authentication. If it matches, the system will allow you to communicate with GE/PE.

Note:

- This permission is intended solely for REST API. It does not enable your application to use SOAP API, Bulk API, Metadata API, etc. for GE/PE.
- This permission is applied only to your application. We do not turn on the API in the GE/PE organization.

Designing Your App to Support Multiple Editions

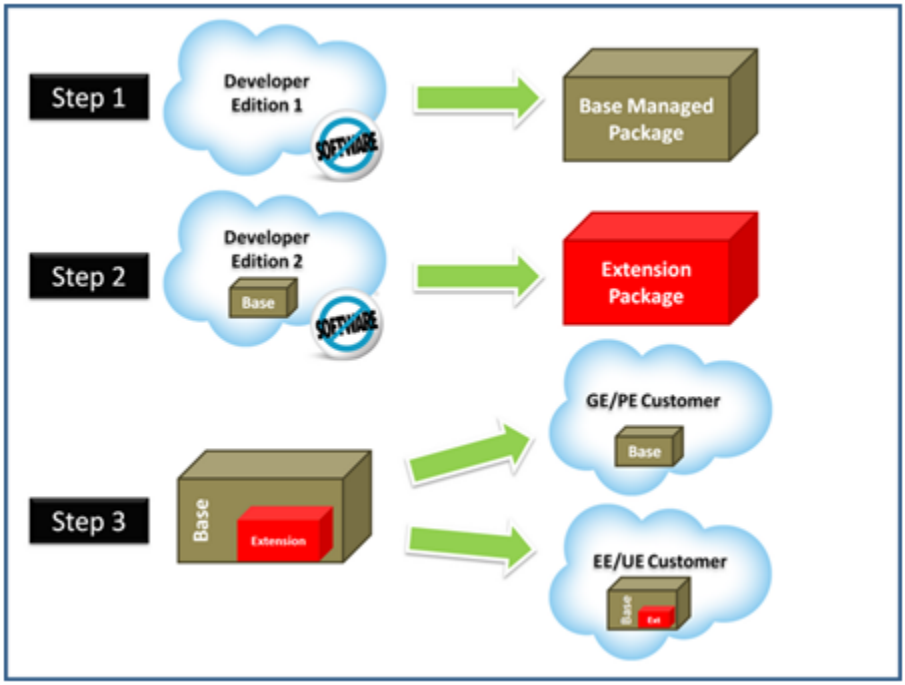
Supporting multiple editions provides the opportunity to release richer versions of your app that can support more advanced features found in EE, UE, and PXE. There are two technologies that can be leveraged to support multiple editions. The first approach uses extension packages and the second leverages Dynamic Apex. There are benefits to both, so be sure to review both strategies before designing your app.

Supporting Multiple Editions Using an Extension Package

This approach uses a base-managed package that contains core app functionality. The base package only contains features supported in Group and Professional Editions. You then use a second managed package, or extension package, that extends and enhances the

base package. The extension package adds more features supported in Enterprise, Unlimited, and Performance Editions. For example, you have a warehouse application that tracks inventory and an extension to this app includes workflow (which isn't available in Group). Your Group and Professional Edition customers can install the base warehouse application, while your other customers install the base package and then the extension package with workflow components.

Using a Base and Extension Package to Support Multiple Editions



Using extension packages enables you to avoid multiple code sets and to upsell your customers. Upgrading a customer only requires installing the extension package.

Here is the process for creating an extension package.

1. Create your base-managed package that uses features supported by Group and Professional Editions.
2. Install your base-managed package in a separate Developer Edition org.
3. In this org, create your extension package that includes more functionality supported in Group and Professional Editions. You can reference the base-managed package to avoid duplicating functionality. Any component that references the base-managed package automatically triggers this package to be an extension package.

Since your extension package depends on your base package, it's important to spend time designing your app and the interfaces between the packages. For example, if the extension package calls an Apex class in the base package, you must make sure that the desired Apex class is made global.

It's also important to consider the entire application life cycle. For example, If you want to add new features, include them in the appropriate package. Ensure that updates to the base package do not break the extension package.

Note: To access history information for custom objects in your extension package, work with the base package owner to enable history tracking in the org for the base package. Enabling history tracking in a base package can result in an error when you install the package and create patch orgs for the extension package.

Supporting Multiple Editions using Dynamic Apex

Using dynamic Apex, dynamic SOQL, and dynamic DML, it's possible to create one managed package for all editions you plan to support without having to use extension packages. Your app behavior can change dynamically based on the features available in your customer's edition. This is useful when designing an app with the intent to support multiple editions.

Make sure that Apex, workflows, etc. in your package do not contain any strongly-typed reference to a feature that isn't supported by GE/PE. This can include adding a custom field on an unsupported standard object, such as Campaigns, or making an Apex reference to features like multi-currency or territory management. When you reference a feature in your package not supported by GE/PE, this package dependency will cause the installation to fail.

Instead, if you use dynamic Apex to first check if these features are available before referencing them, you can install your managed package in GE/PE. The important piece to consider is you must code your Dynamic Apex in a way that can support both use cases. This ensures that if your customer doesn't have a specific feature or object, your app will still function.

Sample Design Scenarios for Group and Professional Editions

Here are some scenarios to help you understand when and how to build for Group and Professional Editions.

Scenario 1: You want to build an app that uses record types

Since record types aren't available in Group Edition, decide if you want to support this edition. Assuming you do, you can build a base-managed package that doesn't include record types. After uploading this managed package in a released state, you can install it into another Developer Edition org to start building the extension. Your extension can add record types that your Professional, Enterprise, Unlimited, and Performance Edition customers can install and use.

Scenario 2: You want to build an app with 80 custom objects

Typically this scenario presents a problem for Group and Professional Edition orgs because of their custom objects limit. However, if you make your app available on the AppExchange, it doesn't count toward custom objects, tabs, and apps limits. So even if your app has 80 custom objects, it installs and works in Group and Professional Edition orgs.

Scenario 3: You want to build an app that makes Apex callouts to a web service

Apex doesn't normally run in Group and Professional Editions. If you get your managed package authorized during the security review, your Apex executes as expected. For this scenario, you build your Apex callout to invoke your external service and then include this class in your package.

Scenario 4: You want to build an app that uses Campaigns

Campaigns are supported by default in Group Edition. For this scenario, you have two options.

- Option 1 - Build a based-managed package that doesn't reference Campaigns. in it's complete, upload, and install it into another Developer Edition org. Build the Campaign functionality as an extension package. Now your Group Edition customers can install the base, while the rest can also install the extension to get extra features.
- Option 2 - This option requires only one package if you use Dynamic Apex as the only reference to Campaigns (as described earlier) and do not include a custom field on the Campaign. Your app can then be installed in Group Edition orgs and higher. If Campaigns is in your customer's edition, then your Dynamic Apex can manipulate Campaigns as expected.

Scenario 5: You want to build a composite app where your receive inbound API calls

You have a separate hosted app that you want to integrate with Salesforce, so you must make API calls to Group and Professional Edition customers. Such calls aren't possible by default. However, if you're an eligible partner, request a special API token that allows your SOAP calls to integrate with Group and Professional Edition orgs. Be sure to embed the Client ID in the SOAP header of your external code.

Connected Apps

USER PERMISSIONS

To read, create, update, or delete connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update all fields except Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND Modify All Data AND Manage Profiles and Permission Sets
To install and uninstall connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To install and uninstall packaged connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps AND Download AppExchange Packages

EDITIONS

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

Connected Apps can be created in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Connected Apps can be installed in: **All** Editions

A connected app integrates an application with Salesforce using APIs. Connected apps use standard SAML and OAuth protocols to authenticate, provide single sign-on, and provide tokens for use with Salesforce APIs. In addition to standard OAuth capabilities, connected apps allow Salesforce admins to set various security policies and have explicit control over who can use the corresponding apps.

Create a Connected App

USER PERMISSIONS

To read, create, update, or delete connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update all fields except Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND Modify All Data AND Manage Profiles and Permission Sets
To install and uninstall connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To install and uninstall packaged connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps AND Download AppExchange Packages

EDITIONS

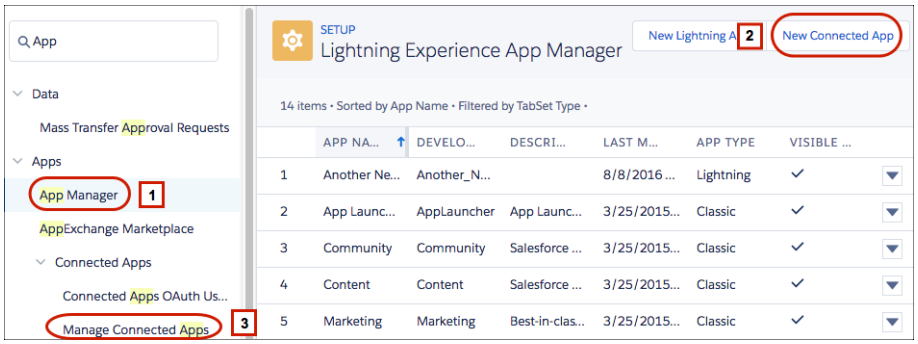
Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Connected Apps can be created in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Connected Apps can be installed in: **All** Editions

When you create a connected app, you specify general information about the app and settings for OAuth, web apps, mobile apps, and canvas apps. You can also customize how the app is invoked by creating a connected app handler with the `ConnectedAppPlugin` Apex class.

The New Connected App wizard walks you through creating a connected app.



- In Lightning Experience, you use the App Manager to create connected apps. From Setup, enter *App* in the Quick Find box, then select **App Manager** (1). Click **New Connected App** (2).
- In Salesforce Classic, from Setup, enter *Apps* in the Quick Find box, then select **Build > Create > Apps**. Under Connected Apps, click **New**.

Note: Be sure to select **App Manager** to create a connected app, not Manage Connected Apps (3).

- Basic Information
- API (Enable OAuth Settings)
- Web App Settings
- Custom Connected App Handler
- Mobile App Settings
- Canvas App Settings

You can create a connected app without specifying authorization, canvas, or mobile settings. This kind of connected app behaves like a "bookmark" to the specified URL that appears in the user's App Launcher and the dropdown app menu. Enter basic information and provide a start URL in the Web App Settings. If the destination requires authentication, the service hosting the destination URL prompts users to provide login credentials when they navigate to it.

When you're finished entering the information, click **Save**. You can now publish your app, make further edits, or delete it. If you're using OAuth, saving your app gives you two new values that the app uses to communicate with Salesforce.

- Consumer Key: A value used by the consumer to identify itself to Salesforce. Referred to as `client_id` in OAuth 2.0.
- Consumer Secret: A secret used by the consumer to establish ownership of the consumer key. Referred to as `client_secret` in OAuth 2.0.

Important: For your convenience, you can update some fields in a connected app and the changes apply immediately to all installed versions of the app. For example, connected app descriptions are immediately updated in each version of the connected app. The following version-independent fields bypass the packaging or installation lifecycle.

- Description
- Info URL
- Logo Image URL

- Callback URL

Basic Information

Specify basic information about your app in this section, including the app name, logo, and contact information.

1. Enter the connected app's name. This name is displayed in the App Manager and on its App Launcher tile.

   Note: The connected app name must be unique for the connected apps in your org. If the connected app was created using the Spring '14 release or later, you can reuse the name of a deleted connected app.

2. Enter the API name used when referring to your app from a program. It defaults to a version of the name without spaces. Only letters, numbers, and underscores are allowed, so if the original app name contains any other characters, edit the default name.
3. Enter the contact email for Salesforce to use when contacting you or your support team. This address isn't given to Salesforce admins who install the app.
4. Enter the contact phone for Salesforce to use in case we want to contact you. This number isn't given to Salesforce admins who install the app.
5. To display your logo on the App Launcher tile, enter a logo image URL. Your logo also appears on the consent page that users see when authenticating. The URL must use HTTPS. Use a GIF, JPG, or PNG file and a file size that's preferably under 20 KB, but at most 100 KB. We resize the image to 128 pixels by 128 pixels, so be sure that you like how it looks. If you don't supply a logo, Salesforce generates one for you using the app's initials.
   - You can upload your own logo image by clicking **Upload logo image**. Select an image from your local file system that meets the size requirements for the logo. When your upload is successful, the URL to the logo appears in the Logo Image URL field. Otherwise, make sure that the logo meets the size requirements.
   - You can also select a logo from the Salesforce samples by clicking **Choose one of our sample logos**. The logos include ones for Salesforce apps, third-party apps, and standards bodies. Click the logo you want, and then copy and paste the URL into the Logo Image URL field.
   - You can use a logo hosted publicly on Salesforce servers by uploading an image as a document from the Documents tab. To get the URL, view the image. Then enter the URL into the Logo Image URL field.
6. If you have a web page with more information about your app, provide an info URL.
7. To display a description on the connected app's App Launcher tile, enter up to 256 characters. If you don't supply a description, just the name appears on the tile.

Note: The App Launcher displays the connected app's name, description, and logo (if provided) on an App Launcher tile. Make sure that the text is meaningful and mistake-free.

API (Enable OAuth Settings)

This section controls how your app communicates with Salesforce. Select **Enable OAuth Settings** to configure authentication settings.

1. If you're setting up OAuth for applications on devices with limited input or display capabilities, such as TVs, appliances, or command-line applications, select **Enable for Device Flow**.

   Note: When enabled, the value for the callback URL defaults to a placeholder, unless you specify your own URL. You can specify your own callback URL as needed, such as when this same consumer is being used for a different flow. A callback URL isn't used in the device authentication flow.

2. Enter the callback URL (endpoint) that Salesforce calls back to your application during OAuth. It's the OAuth redirect URI.

Depending on which OAuth flow you use, the URL is typically the one that a user's browser is redirected to after successful authentication. Because this URL is used for some OAuth flows to pass an access token, the URL must use secure HTTPS or a custom URI scheme. If you enter multiple callback URLs, at run time Salesforce matches the callback URL value specified by the app with one of the values in Callback URL. It must match one of the values to pass validation. Separate multiple callback URLs with line breaks.

The callback URL field has a limit of 2000 characters, cumulatively. If you enter several URLs and they exceed this limit, create another connected app to manage more callback URLs.

3. If you're using the JWT OAuth flow, select **Use Digital Signatures**. If the app uses a certificate, click **Choose File** and browse your system for the certificate.
4. Under Selected OAuth Scopes, add all supported OAuth scopes to. These scopes refer to permissions the user gives to the connected app while it's running. The OAuth token name is in parentheses.

   Access and manage your Chatter feed (chatter_api)
   Allows access to Chatter REST API resources only.

   Access and manage your data (api)
   Allows access to the logged-in user's account using APIs, such as REST API and Bulk API. This value also includes chatter api, which allows access to Chatter REST API resources.

   Access your basic information (id, profile, email, address, phone)
   Allows access to the Identity URL service.

   Access custom permissions (custom_permissions)
   Allows access to the custom permissions in an org associated with the connected app. It shows whether the current user has each permission enabled.

   Allow access to your unique identifier (openid)
   Allows access to the logged-in user's unique identifier for OpenID Connect apps.

   Full access (full)
   Allows access to the logged-in user's data, and encompasses all other scopes. Full doesn't return a refresh token. You must explicitly request the refresh_token scope to get one.

   Perform requests on your behalf at any time (refresh_token, offline_access)
   If the app is eligible to receive a refresh token, allows one to be returned. This scope lets the app interact with the user's data while the user is offline. The refresh_token scope is synonymous with offline_access.

   Provide access to custom applications (visualforce)
   Allows access to Visualforce pages.

   Provide access to your data via the Web (web)
   Allows use of the access_token on the web. It includes Visualforce, which allows access to Visualforce pages.

5. If you're setting up OAuth for a client app that can't keep the client secret confidential and it must use the web server flow, deselect **Require Secret for Web Server Flow**. We still generate a client secret for your app, but this setting instructs the web server flow not to require the `client_secret` parameter in the access token request. If your app can use the user agent flow, we recommend user agent as a more secure option than web server flow without the secret.
6. To authorize a single connected app to introspect all access and refresh tokens within the entire org, select **Introspect all tokens**. By default, all connected apps can introspect their own tokens. In addition, an OAuth client that registers child OAuth 2.0 connected apps through the dynamic client registration endpoint can check the tokens for itself and its child apps. See OpenID Connect Token Introspection Endpoint.
7. To control how the OAuth request handles the ID token, select **Configure ID token**. If the OAuth request includes the **Allow access to your unique identifier (openid)** scope, the returned token can include the ID token.
   - The ID token is always included in access token responses.

- With the primary ID token setting enabled, configure the secondary settings that control the ID token contents in both access and refresh token responses. Specify these settings.

  Token Valid for
  The length of time that the ID token is valid for after it's issued. The period can be from 1 to 720 minutes. The default is 2 minutes.

  ID Token Audiences
  The intended consumers of the ID token. For example, the target service where you use the ID token, such as `https://your_service.com`.

  Include Standard Claims
  Include the standard claims that contain information about the user, such as the user's name, profile, phone number, and address. The OpenID Connect specifications define a set of standard claims to be returned in the ID token.

  Include Custom Attributes
  If your app has specified custom attributes, include them in the ID token.

  Include Custom Permissions
  If your app has specified custom permissions, include them in the ID token.

8. To configure asset token settings if you're setting up your app to issue asset tokens for connected devices, select **Enable Asset Tokens**.
   - Specify these settings.

     Token Valid for
     The length of time that the asset token is valid after it's issued.

     Asset Signing Certificate
     The self-signed certificate that you've already created for signing asset tokens.

     Asset Audiences
     The intended consumers of the asset token. For example, the back-end service for your connected device, such as `https://your_device_backend.com`.

     Include Custom Attributes
     If your app has specified custom attributes, include them in the asset token.

     Include Custom Permissions
     If your app has specified custom permissions, include them in the asset token.

   - Specify the callback URL (endpoint). For example, `https://your_device_backend.com/callback`.
   - Make sure that you select these OAuth scopes, which are required for asset tokens.
     - **Access and manage your data (api)**
     - **Allow access to your unique identifier (openid)**

9. To allow users to be automatically logged out of the connected app service provider when they log out of Salesforce as an identity provider, select **Enable Single Logout**.
10. If you selected **Enable Single Logout**, enter a single logout URL, which is where Salesforce sends logout requests when users log out of Salesforce. The single logout URL must be an absolute URL starting with https://.

If your org had the **No user approval required for users in this organization** option selected on your remote access before the Spring '12 release, users in the org are approved for the app. This option is selected to indicate the approval.

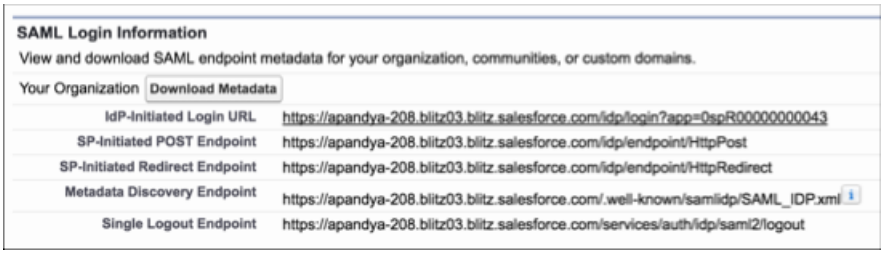
For connected apps, after you've created an app, we recommend that admins install the app, and then set Permitted Users to **Admin-approved users**. If the remote access option wasn't originally selected, the option doesn't show up.

Web App Settings

This section controls your app's web settings.

1. If you want to direct users to a specific location after they've authenticated, enter your app's start URL. If you don't enter a start URL, users are sent to the app's default start page after authentication completes. If the connected app that you're creating is a canvas app, skip this field. The Canvas App URL field contains the URL that gets called for the connected app.
2. If your connected app uses a SAML service provider, select **Enable SAML**. Enter the entity ID, ACS URL, subject type, name ID format, and issuer, available from your service provider.
3. To allow users to be logged out of the connected app service provider when they log out of Salesforce as an identity provider, select **Enable Single Logout**.

4. If you selected **Enable Single Logout**, enter a single logout URL, which is the endpoint where Salesforce sends logout requests when users log out of Salesforce. The single logout URL must be an absolute URL starting with https://.
5. Provide your SP with the Salesforce IdP SLO endpoint. The endpoint is listed in your SAML Login Information as the Single Logout Endpoint. It's also listed in the SAML Metadata file as the Discovery Endpoint. The format for the endpoint is `https://<domain>.my.salesforce.com/services/auth/idp/saml2/logout`, where `<domain>` is your org's My Domain name.



6. Select the HTTP binding type for single logout. Your SP provides this information.
7. Under SAML Service Provider Settings, select **Enable Single Logout**.
8. If the service provider gave you a security certificate, select **Verify Request Signatures**. Browse your system for the certificate and upload it. The certificate is only necessary if you plan to initiate logging in to Salesforce from the service provider and the service provider signs its SAML requests.

   Important: If you upload a certificate, all SAML requests must be signed. If no certificate is uploaded, all SAML requests are accepted.

9. Optionally, select **Encrypt SAML Response** to browse your system for the certificate and upload it. Select an encryption method for encrypting the assertion. Valid encryption algorithm values are AES–128 (128–bit key), AES–256 (256–bit key), and Triple-DES (Triple Data Encryption Algorithm).

Custom Connected App Handler

Write a custom connected app handler in Apex to customize the behavior of the connected app. Create a class that extends the `ConnectedAppPlugin` Apex class, and associate it with the connected app. The class can support new authentication protocols or respond to user attributes in a way that benefits a business process.

1. For Apex Plugin Class, enter the name of the Apex class you created to customize the behavior of the connected app.
2. For Run As, select the name of the user to run the plug-in.

   The plug-in runs on behalf of a user account. If the user isn't authorized for the connected app, use the `authorize` method. For more information, see the `ConnectedAppPlugin` class in the Apex Code Developer's Guide.

Mobile App Settings

Enter your mobile app settings in this section.

1. Enter the mobile start URL to direct users to a specific location when the app is accessed from a mobile device. If you don't enter a mobile start URL, users are sent to the start URL defined under Web App Settings. If the connected app you're creating is a canvas app, you can skip this field. The Canvas App URL field contains the URL that gets called for the connected app.
2. Select **PIN Protect** if your app supports PIN protection. This setting allows the admin to set the session timeout and PIN length for mobile applications after installing the connected app. Salesforce Mobile SDK (https://developer.salesforce.com/page/Mobile_SDK) supports PIN protection. You can also implement it manually by reading the `mobile_policy` object from the user's Identity URL.
3. For App Platform, specify the app platform.
4. For Restrict to Device Type, specify the supported device form factors for the mobile app. If the app supports all form factors, don't choose a value.
5. For App Version, enter the version number of the mobile app.
6. For Minimum OS version, enter the version required for the app.
7. To confirm that this app is only for internal (non-public) distribution, select **Private App** . This setting is required because Apple doesn't allow distribution of public mobile apps outside its App Store.
8. If the mobile app is private, specify the location of the Mobile App Binary file. The format of the file is IPA for iOS, and APK for Android.
9. For iOS apps only:
   a. Specify the location of the Application Icon that is displayed while the app is being downloaded and installed on an iOS device.
   b. Specify the iOS Bundle Identifier.

      Note: For iOS 7 and later, use the same bundle identifier that you used when developing the app in XCode. If you don't, users see two app icons during installation.
10. If the mobile connected app is a public app and you haven't uploaded its binary file to Salesforce, enter the app binary URL.

Note: If you remove mobile integration from a new version of an existing connected app, mobile integration is no longer included in any version of it. For example, you publish a package containing version 1.0 of your connected app with mobile integration. You then remove mobile integration from the app, repackage it, and publish it as version 1.1. If a customer installs the earlier package with version 1.0, the connected app doesn't contain mobile integration.

If you meet the following requirements, your connected app can receive push notifications.

- Your app is built with the Salesforce Mobile SDK.
- Your app implements the Mobile SDK push notification protocol for your platform.

- You're a registered developer with the mobile platform provider (Apple or Google).
- Your app is registered with Apple Push Notification Service (APNS) for iOS push notifications or with Google Cloud Messaging (GCM) for Android push notifications.
- You've implemented Apex handlers for push notifications.

Note: A push-enabled connected app can support only one mobile platform. To support push notifications on Android and iOS versions of your mobile app, create a connected app for each platform.

For details, see the *Salesforce Mobile Push Notifications Implementation Guide*.

To configure push notifications for APNS (iOS):

1. For App Platform, select **iOS**.
2. Select **Push Messaging Enabled**. The dialog box expands to show more settings.
3. For Platform, select **Apple**. The dialog box expands to show more settings.
4. Upload the Signing Key from your Apple developer account.
5. Enter the Key Identifier from your Apple developer account.
6. Enter the Team Identifier from your Apple developer account.

Note: You can use authentication tokens, push certificates, or both. If a token is provided, Salesforce always uses the token. We also use Application Bundle ID when it is provided. We recommend using Application Bundle ID to avoid problems with your push notification settings.

To configure push notifications for GCM (Android):

1. Select **Push Messaging Enabled**.
2. For Supported Push Platform, select **Android GCM**.
3. For Key for Server Applications (API Key), enter the key that you obtained during developer registration with Google.

To change the mobile platform that you've configured for push notifications:

1. Deselect **Push Messaging Enabled**.
2. Save the connected app, and then click **Edit**.
3. To reflect the new platform, change **App Platform** and the associated values in Mobile Settings.
4. Reconfigure push notifications for the new platform.

Canvas App Settings

Two types of canvas apps are available.

- Canvas apps that the org's Salesforce admin installed.
- Canvas personal apps that users installed across orgs. Users access a canvas personal app from the Chatter tab, and are prompted to allow the app to connect to their Salesforce data. Users can choose to make an app a canvas personal app. For more information, see "Canvas Personal Apps" in the *Canvas Developer Guide*.

1. If your connected app is exposed as a canvas app, select **Canvas**.
2. Enter the canvas app URL to the third-party app. The user is directed to this URL when clicking the link to your canvas app.
3. Select an access method. This method specifies how the canvas app initiates the OAuth authentication flow.
   - Signed Request (POST)—OAuth authentication is used, but when Salesforce admins install the canvas app, they implicitly allow access for users. Users aren't prompted to allow apps to access their user information. When you use this access method, authentication is posted directly to the canvas app URL.

If your canvas app uses signed request authentication, don't select **Perform requests on your behalf at any time** for the Selected OAuth Scopes.

- OAuth Webflow (GET)—OAuth authentication is used, and the user is prompted to allow apps to access their information. When you use this access method, the canvas app must initiate the OAuth authentication flow.

4. If you're using SAML single sign-on (SSO) for canvas app authentication, select the **SAML Initiation Method** field. This field is enabled if you select **Enable SAML** in the Web App Settings section. The options for this field include the following.
   - Identity Provider Initiated—Salesforce makes the initial request to start the SSO flow.
   - Service Provider Initiated—Canvas app starts the SSO flow after the app is invoked.
5. Under Locations, select where the canvas app appears to users.
   - Chatter Feed—Canvas app appears in the feed. If selected, create a CanvasPost feed item and ensure that the current user has access to the canvas app.
   - Chatter Tab—Canvas app appears in the app navigation list on the Chatter tab. If selected, the canvas app appears automatically.
   - Console—Canvas app appears in the footer or sidebars of the Salesforce console. If selected, you must choose where the canvas app appears in a console by adding it as a custom console component.
   - Layouts and Mobile Cards—Canvas app can appear on a page layout or a mobile card. If selected, choose where the canvas app appears by adding it to the page layout.
   - Mobile Nav—Canvas app is accessible from the navigation menu in the Salesforce app.

     Note: Canvas apps don't appear in the app navigation menu in Salesforce for Android. To see canvas apps in the navigation menu, log in to Salesforce mobile web.

   - Open CTI—Canvas app appears in the call control tool. If selected, specify the canvas app in your call center's definition file for it to appear.
   - Publisher—Canvas app appears in the publisher. If selected, create a canvas custom quick action and add it to the global layout or to an object layout.
   - Visualforce Page—Canvas app can appear on a Visualforce page. If you add an `<apex:canvasApp>` component to expose a canvas app on a Visualforce page, be sure to select this location for the canvas app. If you don't, you receive an error.
6. To create a global action for your canvas app, select **Create Actions Automatically**. To do so, under Location, select **Publisher**. If you don't use this option, global actions aren't created. You can also create the action later.
7. If you implement your own `Canvas.CanvasLifecycleHandler` Apex class, provide the class name in `Lifecycle Class`. Providing a `Canvas.CanvasLifecycleHandler` Apex class lets you customize context information and add custom behavior to your canvas app.
8. To let users install your app, select **Enable as a Canvas Personal App**. Chatter Tab is the only location that supports canvas personal apps. For details, see "Canvas Personal Apps" in the *Canvas Developer Guide*.

   Note: If you don't see the **Enable as a Canvas Personal App** setting, the admin for the app's destination org hasn't enabled canvas personal apps. For details, see "Enabling Canvas Personal Apps within an Organization" in the *Canvas Developer Guide*.

Edit, Reconfigure, or Delete a Connected App in Salesforce Classic

USER PERMISSIONS

To read, create, update, or delete connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update all fields except Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND Modify All Data AND Manage Profiles and Permission Sets
To install and uninstall connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To install and uninstall packaged connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps AND Download AppExchange Packages

EDITIONS

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

Connected Apps can be created in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Connected Apps can be installed in: **All** Editions

After creating a connected app, you can edit, reconfigure, or delete it. You can restrict access and add custom attributes.

Edit a Connected App

You can edit a connected app at any time to change its description, logo, icon, or callback URL. You can update settings for OAuth, web apps, mobile apps, or canvas apps.

1. Open the list of apps. From Setup, enter `Apps` in the Quick Find box, then select **Apps**.
2. Locate the connected app in the apps list, and click **Edit**.
3. Make your changes, and click **Save**.

Important: For your convenience, you can update some fields in a connected app and the changes apply immediately to all installed versions of the app. For example, connected app descriptions are immediately updated in each version of the connected app. The following version-independent fields bypass the packaging or installation lifecycle.

- Description
- Info URL
- Logo Image URL
- Callback URL

Restrict Access to Trusted IP Ranges and Apply IP Relaxation Ranges

After you create a connected app, you can specify the allowed IP ranges from which a user can log in.In addition, you can apply IP relaxation to all OAuth client apps, including apps that use OAuth and SAML for single sign-on.

Note: You can't relax IP restrictions with the OAuth 2.0 JWT bearer token flow.

To set the trusted IP range:

1. Open the list of apps. From Setup, enter `Apps` in the Quick Find box, then select **Apps**.
2. Locate the connected app in the apps list, and click its name.
3. In the Trusted IP Range for OAuth Web server flow section, click **New**.
4. For the start IP address, enter a valid IP address. For the end IP address, enter the same or higher IP address.

   Enter multiple discontinuous ranges by clicking **New**.

You can give specific users access to an OAuth connected app from outside of the Trusted IP Range. For example, to give users access when traveling, set the connected app to **Relax IP Restrictions for activated devices**. When users try to use the connected app from outside the range with a new browser or device, they are prompted to verify their identity.

To apply IP relaxation to specified IP ranges:

1. Open the list of apps. From Setup, enter `Apps` in the Quick Find box, then select **Apps**.
2. Locate the connected app in the apps list, and click **Manage**.
3. Click **Edit Policies**.
4. In the IP Relaxation field, select **Relax IP Restrictions** or **Relax IP Restrictions for activated devices**.

Note: If **Enforce login IP ranges on every request** is enabled, it affects the IP relaxation behavior. For more information, see Connected App IP Relaxation and Continuous IP Enforcement on page 94.

Add Custom Attributes

After you've created the connected app, you can add custom attributes. With custom attributes, you can get more information about a user's identity, like an address or job title. Custom attributes specify SAML metadata or specify OAuth parameters that are read at OAuth runtime.

1. Open the list of apps. From Setup, enter `Apps` in the Quick Find box, then select **Apps**.
2. Locate the connected app in the apps list, and click its name.
3. Under Custom Attributes, click **New**.

   Each custom attribute must have a unique key and use fields available from the Insert Field menu. For example, assign a key name, such as `country` and insert the field `$Organization.Country`. When using SAML, attributes are sent as SAML attribute statements. When using OAuth, attributes are available as a custom attributes object in the user's Identity URL.

Delete a Connected App

Note: The connected app name must be unique for the connected apps in your org. If the connected app was created using the Spring '14 release or later, you can reuse the name of a deleted connected app.

1. Open the list of apps. From Setup, enter `Apps` in the Quick Find box, then select **Apps**.
2. Locate the connected app in the apps list, and click its name.
3. Click **Delete**, and click **Delete** again to confirm.

When you remove a connected app that is a component of a package, the app remains available until you update the package with a new version. 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 cannot 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.

Install a Connected App

USER PERMISSIONS

To read, create, update, or delete connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update all fields except Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND Modify All Data AND Manage Profiles and Permission Sets
To install and uninstall connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To install and uninstall packaged connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps AND Download AppExchange Packages

EDITIONS

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

Connected Apps can be created in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Connected Apps can be installed in: **All** Editions

You install a connected app in your org by installing a managed package that includes the connected app as a component. For example, ISVs package connected apps to make them available on AppExchange. You can also install an OAuth connected app from the Connected Apps OAuth Usage page. This page lists all OAuth connected apps that users in your org are connecting to currently—including apps that developers created in other Salesforce orgs.

When an Install button appears next to a connected app, users in your org are connecting to the app but it isn't installed in your org. To manage an app's security policies, such as which users can access the app and for how long, you must install the connected app in your org.

1. From Setup, enter `OAuth` in the Quick Find box, then select **Connected Apps OAuth Usage**.
2. Select an app and click **Install**.
3. Click **Manage App Policies** to get details about the app.
4. Click **Edit Policies** to control the app's security policies.

View Connected App Details

USER PERMISSIONS	
To read, create, update, or delete connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update all fields except Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND Modify All Data AND Manage Profiles and Permission Sets
To install and uninstall connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To install and uninstall packaged connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps AND Download AppExchange Packages

EDITIONS

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

Connected Apps can be created in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Connected Apps can be installed in: **All** Editions

The Connected App Detail page provides information about the connected app, including its version, OAuth policies, and scopes (permissions given by the user running the app). You can edit and check usage of the connected app and associate profiles and permissions.

- Click **Edit Policies** to open the Connected App Edit page to change the app configuration, such as Start URLs, Permitted Users, and Refresh Token policies.
- For connected apps that use SAML and if your org is an Identity Provider, click **Download Metadata** to get the service provider SAML login URLs and endpoints that are specific to your community or custom domain configuration. This button appears only if your org is enabled as an Identity Provider, and only with connected apps that use SAML. Instead of downloading metadata, you can access the metadata via a URL in Metadata Discovery Endpoint. Your service provider uses this URL to configure single sign-on to connect to Salesforce.
- Click **View OAuth Usage** to see which OAuth connected apps users are actively connecting to. These apps have an active access or refresh token.
- Click **Enable User Provisioning** to enable user provisioning for a connected app. When enabled, use the User Provisioning wizard to configure or update the settings. After you run the User Provisioning wizard, you can individually manage the linkage between user accounts and their account settings on the third-party system in the User Accounts section.
- Click **Manage Profiles** to select the profiles for the app from the Application Profile Assignment page. Select the profiles to have access to the app (except in Group Edition).

  Important: This option won't appear if the OAuth policy for **Permitted Users** is set to `All users may self-authorize` because this option isn't needed when users can authorize themselves.

- Click **Manage Permission Sets** to select the permission sets for the profiles for this app from the Application Permission Set Assignment page. Select the permission sets to have access to the app.

  Important: This option won't appear if the OAuth policy for **Permitted Users** is set to `All users may self-authorize` because this option isn't needed when users can authorize themselves.

- Click **New** in Service Provider SAML Attributes to create attribute key/value pairs. You can also edit or delete existing attributes.

If you select the **Permitted Users** policy, **Admin-approved users**, then only users whose permissions specifically approve the connected app can run it. If you select **All Users**, profiles and permission sets are ignored.

Manage a Connected App

USER PERMISSIONS

To view the Connected Apps page:	View Configuration
To read, create, update, or delete connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update all fields, except Profiles, Permission Sets, and Service Provider SAML attributes:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update Profiles, Permission Sets, and Service Provider SAML attributes:	Customize Application AND Modify All Data
To install and uninstall connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To install and uninstall packaged connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps AND Download AppExchange Packages

EDITIONS

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

Connected Apps can be created in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Connected Apps can be installed in: **All** Editions

The Connected Apps page lists all connected apps created or installed in the org from other orgs or from App Exchange. You can select an app to get more information and monitor its usage. You can also edit policies, for example, to specify a start URL, add your own connected apps handler, or enable user provisioning.

1. To view and update properties of a connected app, from Setup, enter `Connected Apps` in the Quick Find box, then select **Connected Apps**.
2. Click the name of the connected app to view its detail page.
3. Click **Edit Policies** to change the app's configuration. For example, specify a start URL, add your own connected app handler, add custom attributes, or enable user provisioning. To tighten user access, under OAuth Policies, set Permitted Users to **Admin-approved users are pre-authorized**.

Note: Sessions refresh automatically between every 15 minutes and 24 hours while a user is in the connected app. The refresh is based on the session timeout value set for your org. The refresh is often undetected by the user.

Connected Apps Installed by Salesforce

Some Salesforce client apps, such as the Salesforce app and Salesforce for Outlook, are implemented as connected apps. Salesforce automatically installs them in your org.

Salesforce distributes these client-type connected apps in two managed packages: one for Salesforce for Android and Salesforce for iOS and one for other mobile apps. The connected apps included in the package change with each release. Salesforce installs the package the first time a user accesses one of the apps.

To install or reinstall the Salesforce app package for connected apps, you can install it from App Exchange.

Salesforce packages appear in Setup under the Installed Packages List.

Installed Packages

Action	Package Name	Publisher	Version Number	Namespace Prefix	Status	Allowed Licenses	Used Licenses	Expiration Date	Install Date
Uninstall	Salesforce Connected Apps	Salesforce.com	1.5	sf_com_apps	Free	N/A	N/A	N/A	10/24/2013 4:54 PM
	Description This package contains Connected Applications for all the officially supported Salesforce client applications such as Touch, Salesforce for Outlook, Sa...								
Uninstall	Salesforce1 and Chatter Apps	Salesforce.com	1.6	sf_chttr_apps	Free	N/A	N/A	N/A	9/25/2013 11:40 PM
	Description This package contains Connected Applications for all the officially supported Salesforce1 and Chatter applications on your desktop and mobile devices!								

Click a package name to see the list of components. The following are some components in the Salesforce Connected Apps package.

Package Components

Action	Name	Parent Object	Type
	Forcecom_IDE		Connected App
	Forcecom_Migration_Tool		Connected App
	Dataloader_Bulk		Connected App
	Dataloader_Partner		Connected App
	Salesforce_Touch		Connected App
	Salesforce_Mobile_Dashboards		Connected App
	Workbench		Connected App
	Salesforce_for_Outlook		Connected App

Note: The Force.com IDE and Ant Migration tool are "wrapper" connected apps that use the SOAP API to connect to Salesforce instead of OAuth, which other types of connected apps do. They still use the connected apps framework to allow or deny users access to the apps in an org.

The following are some components for the Salesforce app and Chatter app packages.

Package Components

Action	Name	Parent Object	Type
	SalesforceA		Connected App
	Salesforce_Chatterbox		Connected App
	Chatter_for_BlackBerry		Connected App
	Chatter_for_iOS		Connected App
	Chatter_Desktop		Connected App
	Chatter_for_Android		Connected App

You can manage installed connected apps from the Connected Apps page.

1. From Setup, enter `Connected Apps` in the Quick Find box, then selected **Connected Apps**.
2. View the list of connected apps. The connected apps that Salesforce installs appear in the list as installed by a managed package. They appear along with your other installed connected apps.

Action	Master Label ↑	Application Version	Permitted Users
Edit	Chatter Desktop	7.0	All users may self-authorize
Edit	Chatter Mobile for BlackBerry	7.0	All users may self-authorize
Edit		1.0	All users may self-authorize
Edit		1.0	Admin approved users are pre-authorized
Edit	Salesforce Files	5.0	All users may self-authorize
Edit	Salesforce1 for iOS	7.0	All users may self-authorize
Edit	Salesforce1/Chatter for Android	7.0	All users may self-authorize
Edit	SalesforceA	1.0	All users may self-authorize
Edit		1.0	All users may self-authorize

Connected Apps Installed from the Connected Apps OAuth Usage Page

In addition to the connected apps installed from managed packages, this list contains OAuth connected apps installed from the Connected Apps OAuth Usage page.

Edit Connected App Behavior

USER PERMISSIONS

To read, create, update, or delete connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update all fields except Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND Modify All Data AND Manage Profiles and Permission Sets
To install and uninstall connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To install and uninstall packaged connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps AND Download AppExchange Packages

EDITIONS

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

Connected Apps can be created in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Connected Apps can be installed in: **All** Editions

Modify connected app settings and permissions to control how it behaves. For example, you can change the refresh token and session-level policies for all connected apps running in your org. You can change OAuth policies for OAuth-enabled connected apps. And you can customize the behavior of a connected app with Apex.

1. Open the list of apps.
   - In Salesforce Classic, from Setup, enter `Apps` in the Quick Find box, select **Apps** (under **Build** > **Create**), then click the name of the connected app.
   - In Lightning Experience, from Setup, enter `Apps` in the Quick Find box, select **App Manager**, click ▾, and then select **Edit**.

Basic Information

Basic Information applies to all connected apps, except canvas apps.

- Start URL—For connected apps that use single sign-on. Set the URL to the page where the user starts the authentication process. This URL also appears in the app menu.
- Mobile Start URL—For mobile connected apps to direct users to a specific location when the app is accessed from a mobile device.

If your app is a canvas app, the connected app ignores the start URL fields. Instead, it uses the canvas app URL specified when the connected app was created.

OAuth Policies

OAuth policies apply if the connected app is OAuth-enabled.

- The Permitted Users policy determines who can run the app.
  - **All Users may self-authorize**—Default. Anyone in the org can authorize the app. Users must approve the app the first time they access it.
  - **Admin-approved users are pre-authorized**—Only users with the appropriate profile or permission set can access the app. These users don't have to approve the app before they can access it. Manage profiles for the app by editing each profile's Connected App Access list. Manage permission sets for the app by editing each permission set's Assigned Connected App list. This setting is not available in Group Edition.

  Warning: If you switch from **All Users may self-authorize** to **Admin-approved users are pre-authorized**, anyone using the app loses access, unless a user's permission authorizes the connected app specifically.

  Note: If users have the Use Any API Client permission, they can access any connected app—even if it's set to **Admin-approved users are pre-authorized**. Be careful when using the Use Any API Client permission. It's intended for a limited number of admins.

- The IP Relaxation policy determines whether a user's access to the app is restricted by IP ranges. A Salesforce admin can choose to enforce or bypass IP restrictions by choosing one of the following options.

  Note: IP restrictions are enforced only if they are configured on a user's profile. The SAML bearer assertion and JWT bearer token flows always enforce IP restrictions regardless of the connected app policy.

  - Enforce IP restrictions—Default. A user running this app is subject to the org's IP restrictions, such as IP ranges, which are set in the user's profile.
  - Enforce IP restrictions, but relax for refresh tokens—A user running this app is subject to the org's IP restrictions, such as IP ranges, which are set in the user's profile. However, after initial login, when later using a refresh token to obtain a new access token, the restrictions are bypassed.
  - Relax IP restrictions for activated devices—A user running this app bypasses the org's IP restrictions when either of these conditions is true.
    - The app has a whitelist of IP ranges and is using the web server OAuth authentication flow. Only requests coming from the whitelisted IPs are allowed.
    - The app has no IP-range whitelist, is using the web server or user-agent OAuth authentication flow, and the user successfully completes identity verification if accessing Salesforce from a new browser or device.
  - Relax IP restrictions—A user running this app is not subject to any org IP restrictions.

  Note: If **Enforce login IP ranges on every request** is enabled, it affects the IP relaxation behavior. For more information, see Connected App IP Relaxation and Continuous IP Enforcement on page 94.

- The Refresh Token policy determines whether a refresh token is provided during authorization to get a new access token. If refresh tokens are provided, users can continue to access the OAuth-enabled connected app without having to reauthorize when the access token expires. Admins limit the lifetime of access tokens with the session timeout value. The connected app exchanges the refresh token with an access token to start a new session. A Salesforce admin can choose one of the following refresh token policies.
  - Refresh token is valid until revoked—Default. The refresh token is used indefinitely, unless revoked by the user or Salesforce admin. You revoke tokens on a user's detail page under OAuth Connected Apps or on the OAuth Connected Apps Usage Setup page.
  - Immediately expire refresh token—The refresh token is invalid immediately. The user can use the current session (access token) already issued, but can't obtain a new session when the access token expires.
  - Expire refresh token if not used for `n`—The refresh token is valid as long as it's been used within a specified amount of time. For example, if set to seven days, and the refresh token isn't exchanged for a new session within seven days, the next attempt to use the token fails. The expired token can't generate new sessions. If the refresh token is exchanged within seven days, the token is valid for another seven days. The monitoring period of inactivity also resets.
  - Expire refresh token after `n`—The refresh token is valid for a fixed amount of time. For example, if the policy states one day, the user can obtain new sessions only for 24 hours.

  The Refresh Token policy is evaluated only during usage of the issued refresh token and doesn't affect a user's current session. Refresh tokens are required only when a user's session has expired or isn't available.

  For example, you set a refresh token policy to expire the token after 1 hour. If a user uses the app for 2 hours, the user isn't forced to reauthenticate after 1 hour. However, the user is required to authenticate again when the session expires and the client attempts to exchange its refresh token for a new session.

If your connected app is a canvas app that uses signed request authentication, be sure to:

- Set Permitted Users to **Admin-approved users are pre-authorized**.
- Set Expire Refresh Tokens to **Immediately expire refresh token**.
- Give users access via profiles and permission sets.

Session Policies

Session policies apply to all connected apps.

- Session Timeout Value—Specifies when access tokens expire to end a user's connected app session. You can control how long a user's session lasts by setting the timeout value for the connected app, user profile, or org's session settings (in that order). If you don't set a value or **None** is selected (the default), Salesforce uses the timeout value in the user's profile. If the profile doesn't specify a timeout value, Salesforce uses the timeout value in the org's Session Settings.
- The current permissions for the connected app are also listed in the org's Session Settings.
- High Assurance session required—Requires users to enter a time-based token when trying to log in to access the app.

Mobile App Settings

Mobile App settings apply to mobile connected app that enforce PIN protection.

- Require Pin after—Specifies how much time can pass while the app is idle before the app locks itself and requires the PIN before continuing. Allowable values are none (no locking), 1, 5, 10, and 30 minutes. This policy is only enforced if a corresponding pin length is configured. Enforcement of the policy is the responsibility of the connected app. Apps written using the Salesforce Mobile SDK can enforce this policy, or the app can read the policy from the UserInfo service and enforce the policy.

  Note: This setting doesn't invalidate a user's session. When the session expires due to inactivity, this policy only requires the user to enter a PIN to continue using the current session.

- Pin Length—Sets the length of the identification number sent for authentication confirmation. The length can be from 4 to 8 digits, inclusive.

Custom attributes are available for all connected app. Developers can set custom SAML metadata or custom OAuth attributes for a connected app. Salesforce admins can delete or edit the attributes or add custom attributes. Attributes deleted, edited, or added by admins override attributes set by developers. For more information, see Edit, Reconfigure, or Delete a Connected App in Salesforce Classic on page 85.

Custom Connected App Handler

Write a custom connected app handler in Apex to customize the behavior of the connected app. Create a class that extends the `ConnectedAppPlugin` Apex class, and associate it with the connected app. The class can support new authentication protocols or respond to user attributes in a way that benefits a business process.

1. For Apex Plugin Class, enter the name of the Apex class you created to customize the behavior of the connected app.
2. For Run As, select the name of the user to run the plug-in.

   The plug-in runs on behalf of a user account. If the user isn't authorized for the connected app, use the `authorize` method. For more information, see the `ConnectedAppPlugin` class in the Apex Code Developer's Guide.

User Provisioning Settings

Enable User Provisioning—Enable user provisioning for the connected app to create, update, and delete user accounts in the connected app based on users in your Salesforce org. Salesforce provides a wizard to guide you through configuring or updating user provisioning settings.

Connected App IP Relaxation and Continuous IP Enforcement

If you relaxed IP restrictions for your OAuth-enabled connected app and your org has **Enforce login IP ranges on every request** enabled, the access to your connected app can change.

This change applies to client access, including mobile devices, for all OAuth-enabled connected apps. IP relaxation does not apply to SAML-enabled connected apps unless they are also OAuth-enabled for single sign-on.

Note: IP restrictions are enforced only if they are configured on a user's profile. The SAML bearer assertion and JWT bearer token flows always enforce IP restrictions regardless of the connected app policy.

Table 1: Connected App IP Relaxation Settings and Continuous IP Enforcement

IP Relaxation	When Continuous IP Enforcement Is Disabled (Default)	When Continuous IP Enforcement Is Enabled
`Enforce IP restrictions`	A user running this app is subject to the org's IP restrictions, such as IP ranges set in the user's profile.	A user running this app is subject to the org's IP restrictions, such as IP ranges set in the user's profile.
`Enforce IP restrictions, but`	A user running this app is subject to the org's IP restrictions, such as IP ranges set in the user's profile, during initial login. However, these	A user running this app is subject to the org's IP restrictions, such as IP ranges set in the user's profile, during initial login. These restrictions are relaxed when the user is later

EDITIONS

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

Connected Apps can be created in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Connected Apps can be installed in: **All** Editions

IP Relaxation	When Continuous IP Enforcement Is Disabled (Default)	When Continuous IP Enforcement Is Enabled
`relax for refresh tokens`	restrictions are relaxed when the app is later using a refresh token to obtain a new access token.	using a refresh token to obtain a new access token. However, the user can't access the following for security reasons: • Change password • Add a time-based token • Any pages in a login flow
`Relax IP restrictions for activated devices`	A user running this app bypasses the org's IP restrictions when either of these conditions is true: • The app has IP ranges whitelisted and is using the web server OAuth authentication flow. Only requests coming from the whitelisted IPs are allowed. • The app has no IP range whitelist, is using the web server or user-agent OAuth authentication flow, and the user successfully completes identity verification if accessing Salesforce from a new browser or device.	A user running this app bypasses the org's IP restrictions when either of the OAuth conditions in the previous column is true. However, the user can't access the following for security reasons: • Change password • Add a time-based token • Any pages in a login flow
`Relax IP restrictions`	A user running this connected app is not subject to any IP restrictions.	A user running this connected app is not subject to any IP restrictions. However, the user can't access the following for security reasons: • Change password • Add a time-based token • Any pages in a login flow

Monitor Usage for an OAuth Connected App

USER PERMISSIONS

To view the OAuth Connected Apps Usage page:	View Setup and Configuration AND Manage Users
To read, create, update, or delete connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update all fields except Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND Modify All Data
To install and uninstall connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps

EDITIONS

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

Connected Apps can be created in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Connected Apps can be installed in: **All** Editions

To install and uninstall packaged connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps AND Download AppExchange Packages

The OAuth Usage page lists all OAuth connected apps that users in the org are connecting to. These apps have an active access or refresh token. If all tokens for an app are revoked or expired, the app disappears from this list.

The Connected Apps OAuth Usage page displays all current OAuth connections to the org, regardless of where the app came from. If an app has an Install button next to it, the connected app was created in another org. Install the app so that you can manage its security policies, for example, which users can access the app and for how long.

To view information about OAuth connected apps, from Setup, enter `OAuth` in the Quick Find box, then select **Connected Apps OAuth Usage**. The resulting list of apps can be long because it contains all Salesforce and custom OAuth apps available to your users, not just the ones installed in your org. For example, it lists apps from AppExchange and Salesforce partners.

Connected App
Name of the connected app. The list contains only the apps that users in the org are using. These apps have an active access or refresh token.

Description
Description of the connected app.

Manage App Policies
Click **Manage App Policies** to open the detail page for the connected app. From the detail page, you can click **Edit Policies** to manage the app's access and security policies.

User Count
Number of users who are using the app. Click a User Count number to open the Connected Apps User's Usage page to see information about users, including:

- When they first used the app
- Most recent time they used the app
- Total number of times they used the app

From this page, you can end a user's access to the current session by clicking **Revoke**. Or, you can click **Revoke All** at the top of the page to log out everyone currently using the app.

Actions
You can perform one of the following actions.

- **Install**—Make the OAuth connected app available for access and security policy management. When you click install, the Manage App Policies link appears next to the app. Click the link to open the app's detail page where you can set policies. Install appears next to apps that were created in another org but you can't manage their policies in your org.
- **Uninstall**—Remove the local copy of the OAuth connected app. Click uninstall only when the original developer deletes the app in the other Salesforce org. Uninstall doesn't remove the connected app. It just removes the local copy that you installed to set the app's OAuth policies in your org. By uninstalling the app, you're only removing the OAuth policies that you set for the app. You're actually loosening your security measures because you're removing the policies for apps that users can still access.
- **Block**—Make the OAuth connected app inaccessible to your org's users. Blocking an app ends all current user sessions and prevents future sessions until you click **Unblock**.
- **Unblock**—Give users access to the connected app. By unblocking the app, users can log in and run the app. If Unblock is disabled, the app is blocked org-wide because it's not whitelisted. To whitelist the app, click **Install**. Then click **Edit Policies**,

and under the app's OAuth settings, set Permitted Users to **Admin approved users are pre-authorized**. You can whitelist apps only if you've asked Salesforce to enable the API client whitelisting feature.

Uninstall a Connected App

USER PERMISSIONS	
To read, create, update, or delete connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update all fields except Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND either Modify All Data OR Manage Connected Apps
To update Profiles, Permission Sets, and Service Provider SAML Attributes:	Customize Application AND Modify All Data AND Manage Profiles and Permission Sets
To install and uninstall connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps
To install and uninstall packaged connected apps:	Customize Application AND either Modify All Data OR Manage Connected Apps AND Download AppExchange Packages

EDITIONS

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

Connected Apps can be created in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Connected Apps can be installed in: **All** Editions

You can remove a connected app from your org in several ways. You can uninstall an OAuth connected app from the Connected App OAuth Usage page. Or, if the connected app was installed as a managed package, you can remove the app by uninstalling the package.

The Connected App OAuth Usage page lists all OAuth connected apps that users in your org are accessing. If the app has an Uninstall button next to it, a developer in another org created it. It's recommended that you uninstall an app only when the original developer deletes the app on the other org. Uninstall doesn't remove the connected app. It just removes the local copy that you installed to set the app's OAuth policies in your org. By uninstalling the app, you're only removing the OAuth policies that you set for the app in your org. You're actually loosening your security measures.

To make the connected app inaccessible to your org's users, click **Block**. Blocking an app ends all current user sessions with the connected app and prevents all new sessions until you click **Unblock**.

Don't uninstall connected app packages owned and distributed by Salesforce, such as the Salesforce app. Salesforce installs and manages these packages.

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.

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.

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.

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.

Environment Hub Best Practices

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

Environment Hub FAQ

Answers to common questions about the Environment Hub.

Considerations for the Environment Hub in Lightning Experience

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

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.

Configure the Environment Hub

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

Enable My Domain for the Environment Hub

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

EDITIONS

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

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

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**.
   b. Create a profile, or edit an existing one.
   c. Edit the profile's settings.

USER PERMISSIONS

To set up and configure the Environment Hub:

- Manage Environment Hub

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.

Profile Section	Environment Hub Settings
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: • 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 orgs to the Environment Hub and create SSO user mappings. Enable My Domain in the org where the Environment Hub is installed.

USER PERMISSIONS

To set up a domain name:
- Customize Application

1. Find an available domain name and sign up for it.
   a. From Setup, enter `My Domain` in the `Quick Find` box, then select **My Domain**.
   b. Enter the subdomain name you want to use within the sample URL.
   c. Select **Check Availability**. If your name is already taken, choose a different one.

d. Select **Register Domain**.
   You receive a confirmation email from Salesforce when your new domain is ready for testing.

2. Test your domain name and deploy it to your org.
   a. Click the URL in the confirmation email to log in to Salesforce using your new domain. Alternatively, from Setup, enter `My Domain` in the `Quick Find` box, select **My Domain**, and then select **Click here to login**.
   b. Test the new domain by clicking tabs and links within your org. Notice that all pages show your new domain name.

      Tip: If you use custom buttons or Visualforce pages in your org, test them before deploying the new domain name. If you used instance-based URLs in your customizations, your links are broken.

   c. To roll out the new domain name to your org, from Setup, enter `My Domain` in the `Quick Find` box, select **My Domain**, and then select **Deploy to Users**.
      The domain is activated immediately, and your users are redirected to pages with the new domain.

3. Set the domain login policy for users accessing your pages.
   a. From Setup, enter `My Domain` in the `Quick Find` box, then select **My Domain**.
   b. Under My Domain Settings, select **Edit**.
   c. To turn off authentication for users who do not use your domain-specific login page, select the login policy. This option enhances security by preventing login attempts by anyone who doesn't know your domain name.
   d. Choose a redirect policy based on the level of security that you want. You have these 3 options, in order of increasing security.
      - Redirect users to the same page within the domain.
      - Redirect users with a warning.
      - Prevent redirecting by having users enter the new domain name.

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

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 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 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.

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

USER PERMISSIONS

To connect an organization to the Environment Hub:

- Connect Organization to Environment Hub

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**.

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.

USER PERMISSIONS

To set up and configure the Environment Hub:

- Manage Environment Hub

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

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.

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.

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.

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.

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.

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.

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 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**.

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.
- 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.

EDITIONS

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

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

Environment Hub FAQ

Answers to common questions about the Environment Hub.

Can I use the Environment Hub in Lightning Experience?

Where do I install the Environment Hub?

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.

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

Can I enable the Environment Hub in a sandbox org?

EDITIONS

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

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

What kinds of orgs can I create in the Environment Hub?
How is locale determined for the orgs I create in the Environment Hub?
Are the orgs that I create in the Environment Hub the same as the ones I created in the Partner Portal?
Can an org be a member of multiple Environment Hubs?
Can I disable the Environment Hub?

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

Org Type	Best Used For	Expires After
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

Developer Hub

The Developer Hub (Dev Hub) lets you create and manage scratch orgs. The scratch org is a source-driven and disposable deployment of Salesforce code and metadata, made for developers and automation. A scratch org is fully configurable, allowing developers to emulate different Salesforce editions with different features and preferences. Scratch orgs are a central feature of Salesforce DX, an open developer experience for developing and managing Salesforce apps across their entire lifecycle.

To work with scratch orgs, you must first enable the Developer Hub (Dev Hub) in your production or business org. You then use the Salesforce DX command-line interface (CLI) to create scratch orgs. You have two ways of managing your scratch orgs: using CLI commands or the Salesforce graphical interface.

EDITIONS

Available in: Lightning Experience

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

Note: Use the Dev Hub to manage scratch orgs. Continue using the Environment Hub to manage other types of orgs, including production and trial orgs

SEE ALSO:

Salesforce DX Setup Guide

Salesforce DX Developer Guide

Scratch Org Allocations for Partners

To ensure optimal performance when using Salesforce DX, partners are allocated a set number of scratch orgs in your business org. These allocations determine how many scratch orgs you can create daily, and how many can be active at a given point.

By default, Salesforce deletes scratch orgs and their associated ActiveScratchOrg records from your Dev Hub when a scratch org expires. All partners get 100 Salesforce Limited Access - Free user licenses.

Premium Partners

- 150 active
- 300 daily

Other Partners

- 40 active

- 80 daily

Partner Trials

- 20 active
- 40 daily

Enable Dev Hub in Your Org

Enable Dev Hub in your org so you can create and manage scratch orgs from the command line. Scratch orgs are disposable Salesforce orgs to support development and testing.

Enabling Dev Hub in a production or business org is completely safe and won't cause any performance or customer issues. Dev Hub comprises objects with permissions that allow admins to control the level of access available to a user and an org.

Consider these factors if you select a trial or Developer Edition org as your Dev Hub.

- You can create up to 6 scratch orgs and package versions per day, with a maximum of 3 active scratch orgs.
- Trial orgs expire on their expiration date.
- Developer Edition orgs that are inactive for 365 days are deactivated.
- You can define a namespace in a Developer Edition org that isn't your Dev Hub, and you can enable Dev Hub in a Developer Edition org that doesn't contain a namespace.
- If you plan to create package versions or run continuous integration jobs, it's better to use a production or business org as your Dev Hub because of higher scratch org and package version limits. Package versions are associated with your Dev Hub org. When a trial or Developer Edition org expires, you lose access to the package versions.

EDITIONS

Available in: Salesforce Classic and Lightning Experience

Dev Hub available in: **Developer**, **Enterprise**, **Performance**, and **Unlimited** Editions

Scratch orgs available in: **Developer**, **Enterprise**, **Group**, and **Professional** Editions

Note: Partner trial orgs signed up from the partner community have different scratch org limits. See Scratch Org Allocations for Partners. Partners can create partner edition scratch orgs: Partner Developer, Partner Enterprise, Partner Group, and Partner Professional. This feature is available only if creating scratch orgs from a Dev Hub in a partner business org. See Supported Scratch Org Editions for Partners in the *ISVforce Guide* for details.

The Dev Hub org instance determines where scratch orgs are created.

- Scratch orgs created from a Dev Hub org in Government Cloud are created on a Government Cloud instance.
- Scratch orgs created from a Dev Hub org in Public Cloud are created on a Public Cloud instance.

To enable Dev Hub in an org:

1. Log in as System Administrator to your Developer Edition, trial, or production org (if you're a customer), or your business org (if you're an ISV).
2. From Setup, enter `Dev Hub` in the Quick Find box and select **Dev Hub**.

   If you don't see Dev Hub in the Setup menu, make sure your org is one of the supported editions.
3. To enable Dev Hub, click **Enable**.

   After you enable Dev Hub, you can't disable it.

Add Salesforce DX Users

System administrators can access the Dev Hub org by default. You can enable more users to access the Dev Hub org so that they can also create scratch orgs.

You can use Salesforce DX with these Standard user licenses: Salesforce, Salesforce Platform, and Salesforce Limited Access - Free.

You can add a user with the System Administrator profile. You can also add a user with a Standard User profile as long as you apply the set of permissions required for Salesforce DX. Because you're adding users to a Dev Hub org, avoid adding them as system administrators unless their work requires that level of authority.

1. Add the user to your Dev Hub org, if necessary.
   a. In Setup, enter `Users` in the Quick Find box, then select **Users**.
   b. Click **New User**.
   c. Fill out the form, and assign the System Administrator or Standard User profile.
   d. Click **Save**.

   If you're adding a System Administrator user, you can stop here.

2. If you're adding a Standard User, create a permission set for Salesforce DX users if you don't have one.
   a. From Setup, enter `Permission Sets` in the Quick Find box, then select **Permission Sets**.
   b. Click **New**.
   c. Enter a label, API name, and description. The API name is a unique name used by the API and managed packages.
   d. Select a user license option. If you plan to assign this permission set to multiple users with different licenses, select **None**.
   e. Click **Save**. The permission set overview page appears. From here, you can navigate to the permissions you want to add or change for Salesforce DX. For the required permissions, see Permission Set for Salesforce DX Users on page 110.

3. Apply the Salesforce DX permission set to the Standard User.
   a. From Setup, enter `Permission Sets` in the Quick Find box, then select **Permission Sets**.
   b. Select the Salesforce DX permission set.
   c. In the permission set toolbar, click **Manage Assignments**.
   d. Click **Add Assignments**.
   e. Select the user to assign the permission set to.
   f. Click **Assign**.
   g. Click **Done**.

You can limit a user's access by modifying the permissions.

Permission Set for Salesforce DX Users

To give full access to the Dev Hub org, the permission set must contain these permissions.

- Object Settings > Scratch Org Infos > Read, Create, Edit, and Delete
- Object Settings > Active Scratch Orgs > Read, Edit, and Delete
- Object Settings > Namespace Registries > Read

To work with second-generation packages in the Dev Hub org, the permission set must also contain:

- System Permissions > Create and Update Second-Generation Packages

This permission provides access to:

Salesforce CLI Command	Tooling API Object (Create and Edit)
`force:package:create`	Package2
`force:package:version:create`	Package2VersionCreateRequest
`force:package:version:update`	Package2Version

SEE ALSO:

Salesforce Help: User Licenses

Free Limited Access License

Looking to use Dev Hub in your production or business org but don't have a Salesforce user license? Look no further. The Salesforce Limited Access - Free license lets developers access Dev Hub to create and manage scratch orgs. In addition to this functionality, you can access Chatter to collaborate with other users.

The main purpose of this license is to enable developers to create scratch orgs. Your Salesforce admin has to grant appropriate permissions to the Dev Hub objects (ScratchOrgInfo, ActiveScratchOrg, and NamespaceRegistry) to get you started. However, Salesforce objects such as Accounts, Contacts, and Opportunities are not accessible via this license.

To give full access to the Dev Hub org, create a permission set that contains these permissions.

- Object Settings > Scratch Org Info > Read, Create, and Delete
- Object Settings > Active Scratch Org > Read and Delete
- Object Settings > Namespace Registry > Read (to use a linked namespace in a scratch org)

For more information, see Add Salesforce DX Users. Contact your Salesforce account representative to request the Free Limited Access License. Salesforce administrators can upgrade a Salesforce Limited Access - Free license to a standard Salesforce license at any time.

Manage Scratch Orgs from Dev Hub

You can view and delete your scratch orgs and their associated requests from the Dev Hub.

In Dev Hub, ActiveScratchOrgs represent the scratch orgs that are currently in use. ScratchOrgInfos represent the requests that were used to create scratch orgs and provide historical context.

1. Log in to Dev Hub org as the System Administrator or as a user with the Salesforce DX permissions.
2. From the App Launcher, select **Active Scratch Org** to see a list of all active scratch orgs.

   To view more details about a scratch org, click the link in the Number column.
3. To delete an active scratch org from the Active Scratch Org list view, choose **Delete** from the dropdown.

   Deleting an active scratch org does not delete the request (ScratchOrgInfo) that created it, but it does free up a scratch org so that it doesn't count against your allocations.
4. To view the requests that created the scratch orgs, select **Scratch Org Info** from the App Launcher.

   To view more details about a request, click the link in the Number column. The details of a scratch org request include whether it's active, expired, or deleted.
5. To delete the request that was used to create a scratch org, choose **Delete** from the dropdown.

   Deleting the request (ScratchOrgInfo) also deletes the active scratch org.

Link a Namespace to a Dev Hub Org

To use a namespace with a scratch org, you must link the Developer Edition org where the namespace is registered to a Dev Hub org.

Complete these tasks before you link a namespace.

- If you don't have an org with a registered namespace, create a Developer Edition org that is separate from the Dev Hub or scratch orgs. If you already have an org with a registered namespace, go to Step 1.
- In the Developer Edition org, create and register the namespace.

  Important: Choose namespaces carefully. If you're trying out this feature or need a namespace for testing purposes, choose a disposable namespace. Don't choose a namespace that you want to use in the future for a production org or some other real use case. Once you associate a namespace with an org, you can't change it or reuse it.

1. Log in to your Dev Hub org as the System Administrator or as a user with the Salesforce DX Namespace Registry permissions.
2. (Required) If you have not already done so, define and deploy a My Domain name.

   Tip: Why do you need a My Domain? A My Domain adds a subdomain to your Salesforce org URL so that it's unique. As part of the Namespace Registry linking process, you'll be logging into two distinct orgs simultaneously (your Dev Hub org and your Developer Edition org), and your browser can't reliably distinguish between the two without a My Domain.

   You receive an email when your domain name is ready for testing. It can take a few minutes.
3. From the App Launcher menu, select **Namespace Registries**.
4. Click **Link Namespace**.

   Tip: Make sure your browser allows pop-ups from your Dev Hub org.
5. Log in to the Developer Edition org in which your namespace is registered using the org's System Administrator's credentials.

   We recommend that you link Developer Edition orgs with a registered namespace to only a single Dev Hub org. You cannot link orgs without a namespace, sandboxes, scratch orgs, patch orgs, and branch orgs to the Namespace Registry.

To view all the namespaces linked to the Namespace Registry, select the **All Namespace Registries** view.

Supported Scratch Org Editions for Partners

Create partner edition scratch orgs from a Dev Hub partner business org.

Supported partner scratch org editions include:

- Partner Developer
- Partner Enterprise
- Partner Group
- Partner Professional

Indicate the partner edition in the scratch org definition file.

```
"edition": "Partner Enterprise",
```

If you attempt to create a partner scratch org and see this error, confirm that you're using an active partner business org. Contact the Partner Community for further assistance.

```
ERROR:  You don't have permission to create Partner Edition organizations.
To enable this functionality, please log a case in the Partner Community.
```

License limits for partner scratch orgs are similar to partner edition orgs created in Environment Hub. Get the details on the Partner Community.

Notifications for Package Errors

EDITIONS

Available in: Salesforce Classic

Accurately track failed package installations, upgrades, and uninstallations in subscriber orgs with the Notifications for Package Errors feature. Proactively address issues with managed and unmanaged packages and provide support to subscribers so that they can successfully install and upgrade your apps.

You can choose to send a notification to an email address in your org when a subscriber's attempt to install, upgrade, or uninstall a packaged app fails. To enable this feature, contact your Salesforce representative.

Errors can happen with these package operations:

- Installation
- Upgrade
- Push upgrade
- Uninstallation

When an installation fails, an email is sent to the specified address with the following details:

- Reason for the failure
- Subscriber org information
- Metadata of the package that wasn't installed properly
- Who attempted to install the package

This example email is for a package installation that failed because the base package wasn't installed before the subscriber tried to install an extension.

```
On Mon, Jul 13, 2015 at 11:51 AM, NO REPLY <no-reply@salesforce.com> wrote:
The install of your package failed. Here are the details:

Error Message: 00DD00000007uJp: VALIDATION_FAILED [DB 0710 DE1 Pkg1 1.2: A required package
 is missing: Package "DB 0710 DE1 Pkg1", Version 1.2 or later must be installed first.]
Date/Time of Occurrence = Mon Jul 13 18:51:20 GMT 2015

Subscriber Org Name = DB 071015 EE 1
Subscriber Org ID = 00DD00000007uJp
Subscriber Org Status = TRIAL
Subscriber Org Edition = Enterprise Edition

Package Name = DB 0710 DE2 Pkg1
Package ID = 033D000000060EE
Package Namespace = DB_0710_DE2
Package Type = MANAGED
Package Version Name = 1.2
Package Version Number = 1.2
Package Version Id = 04tD00000006QoF

Installer Name = Admin User
Installer Email Address = dburki@salesforce.com
```

Set the Notification Email Address

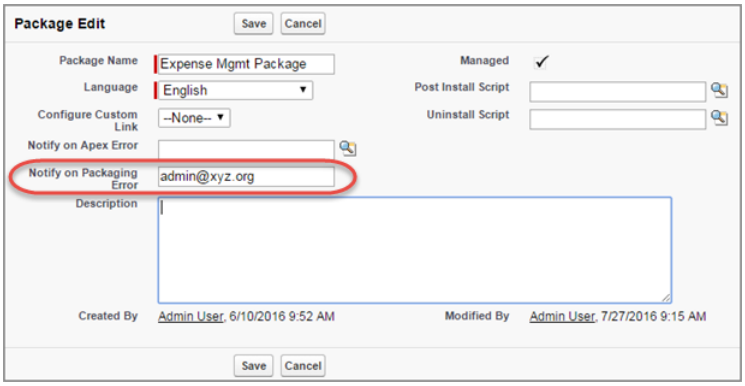
Specify which address to email when a package installation, upgrade, or uninstallation fails.

Notifications are sent only for package versions that are uploaded after the address is added. For example, if you upload package version 1.0 and then set the notification address, notifications aren't sent for failures related to version 1.0. Notifications start when version 2.0 is uploaded.

Also, you can't change or remove the notification email address for the package after it's been uploaded.

1. To enable this feature, contact your Salesforce representative.
2. From Setup, enter `Packages` in the `Quick Find` box, then select **Packages**.
3. Click the package name, and then click **Edit** on the package detail page.
4. Enter the email address to send notifications to, and click **Save**.

Notifications for Package Errors Configured in a Partner Org



CHAPTER 5 Package and Test Your Solution

In this chapter ...

- About Managed Packages
- Installing a Package
- Uninstalling a Package
- Installing Managed Packages Using the API
- Resolving Apex Test Failures
- Running Apex on Package Install/Upgrade
- Running Apex on Package Uninstall
- Publishing Extensions to Managed Packages

Learn how to package, upload, and install a beta version of your solution as part an iterative development approach. After your beta is up and running, learn how to test, fix, extend, and uninstall the solution.

About Managed Packages

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.

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.

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*.

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

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. Under Create, select **Packages**.

   Note: This item is only available in Salesforce Classic.

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**.

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.

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

What are Beta Versions of Managed Packages?

A beta package is an early version of a managed package that is uploaded in a Managed - Beta state. 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.

Before installing a beta version of a managed package, review the following notes:

- Beta packages can be installed in sandbox or Developer Edition organizations, or test organizations furnished through the Environment Hub only.
- The components of a beta package are editable by the developer's organization until a Managed - Released package is uploaded.
- Beta versions aren't considered major releases, so the package version number doesn't change.
- Beta packages are not upgradeable. Because developers can still edit the components of a beta package, the Managed - Released version might not be compatible with the beta package installed. Uninstall the beta package and install a new beta package or released version. For more information, see Uninstalling a Package on page 128 and Installing a Package on page 125.

Creating and Uploading a Beta Package

Use the following procedure to create and upload a beta package through the UI. (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*.)

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, in the `Notify on Apex Error` field, enter the username of the person who should receive an email notification if an exception occurs in Apex code that is not caught by the code. If you don't specify a username, all uncaught exceptions generate an email notification that is sent to Salesforce.
   f. 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.

g. Optionally, choose a custom link from the `Configure Custom Link` field to display configuration information to installers. The custom link displays as a **Configure** link within Salesforce on the Installed Packages page and package detail page of the subscriber's organization.

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. This is an Apex script that runs in the subscriber organization after the package is installed or upgraded. For more information, see Running Apex on Package Install/Upgrade on page 130.

j. Optionally, specify an uninstall script. This is an Apex script that runs in the subscriber organization after the package is uninstalled. For more information, see Running Apex on Package Uninstall on page 134.

k. On the right side of the screen, select the **Managed** checkbox.

l. Click **Save**.

2. Optionally, change the API access privileges. By default, API access is set to `Unrestricted`, but you can change this setting to further restrict API access of the components in the package.

3. Add the necessary components for your app.

   a. Click **Add Components**.

   b. From the drop-down list, choose the type of component.

   c. Select the components you want to add.

   Note: Some components cannot be added to Managed - Released packages. For a list of packageable components, see Components Available in Managed Packages on page 29. If you add S-Controls and documents, keep in mind that their combined size must be less than 10 MB. Also, 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 though they might not 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, see Components Automatically Added to Packages on page 44.

4. Optionally, click **View Dependencies** and review a list of components that rely on other components, permissions, or preferences within the package. For more information on dependencies, see About Dependencies on page 56. Click **Done** to return to the Package detail page.

5. Click **Upload**.

6. On the Upload Package page, do the following:

   a. Enter a `Version Name`, such as *`Spring 11 — Beta`*.

   b. Enter a `Version Number`, such as *`1.0`*. All beta packages use the same version number until you upload a Managed - Released package.

   c. Select a `Release Type` of Managed - Beta.

   Note: Beta packages can only be installed in Developer Edition, sandbox, or test organizations requested through the Environment Hub, and thus can't be pushed to customer organizations.

   d. 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.

e. 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.

f. Click **Upload**.

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

Create and Upload a Managed Package

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:

- Determine if you want to create and upload a managed or unmanaged package.
- 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 on page 116. Your developer settings specify your namespace prefix on page 117, the Salesforce organization where you install the License Management Application (LMA), and the unmanaged package you want to convert into a managed package.

Use the following procedure to create and upload a managed package through the UI. (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*.)

This procedure assumes you have already created a namespace and beta package. If you're uploading a beta package for testing, see Creating and Uploading a Beta Package on page 118.

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, in the `Notify on Apex Error` field, enter the username of the person who should receive an email notification if an exception occurs in Apex code that is not caught by the code. If you don't specify a username, all uncaught exceptions generate an email notification that is sent to Salesforce.
   f. 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.
   g. Optionally, choose a custom link from the `Configure Custom Link` field to display configuration information to installers. The custom link displays as a **Configure** link within Salesforce on the Installed Packages page and package detail page of the subscriber's 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

   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. This is an Apex script that runs in the subscriber organization after the package is installed or upgraded. For more information, see Running Apex on Package Install/Upgrade on page 130.
   j. Optionally, specify an uninstall script. This is an Apex script that runs in the subscriber organization after the package is uninstalled. For more information, see Running Apex on Package Uninstall on page 134.
   k. On the right side of the screen, select the **Managed** checkbox.
   l. Click **Save**.
2. Optionally, change the API access privileges. By default, API access is set to `Unrestricted`, but you can change this setting to further restrict API access of the components in the package.
3. Add the necessary components for your app.
   a. Click **Add Components**.
   b. From the drop-down list, choose the type of component.
   c. Select the components you want to add.

      Note: Some components cannot be added to Managed - Released packages. For a list of packageable components, see Components Available in Managed Packages on page 29. If you add S-Controls and documents, keep in mind that their combined size must be less than 10 MB. Also, 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 though they might not 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, see Components Automatically Added to Packages on page 44.

4. Optionally, click **View Dependencies** and review a list of components that rely on other components, permissions, or preferences within the package. For more information on dependencies, see About Dependencies on page 56. Click **Done** to return to the Package detail page.
5. Click **Upload**.
6. On the Upload Package page, do the following:
   a. Enter a `Version Name`, such as *`Spring 12`*. The version name is the marketing name for a specific release of a package and allows you to create a more descriptive title for the version than just a number.
   b. Enter a `Version Number`, such as *`1.0`*. For more information on versions, see Update Your Solution on page 274.
   c. Select a `Release Type` of Managed - Released.
   d. Change the `Description`, if necessary.
   e. 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.

   f. 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.

g. 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.

h. Click **Upload**.

7. Once 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."

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 Details on page 123
- Components on page 124
- Versions on page 124
- Patch Organizations on page 125

From the Package Detail page, you can do any of the following:

- 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 are no longer bundled together within this package.
- Click **Upload** to upload the package. You are notified by email when the upload is complete.
- 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`.

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 don't 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.
Notify on Packaging Error	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.
Package Name	The name of the package, given by the publisher.
Push Upgrade Exclusion List	A comma-separated list of org IDs to exclude when you push a package upgrade to subscribers.
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 on page 130.
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 on page 134.

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.

Note: Some related components are automatically included in the package even though they might not 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 Components Automatically Added on page 44.

Package components frequently depend on other components that aren't always added to the package explicitly. Each time you change a package, Salesforce checks for dependencies and displays the components as package members. Package Manager automatically checks for dependencies and shows the component relationship to the package in the Include By column of the Package Details.

When your package contains 1,000 or more components, you can decide when to refresh the components list in the Package Details and avoid a long wait while this page loads. The components list refreshes automatically for packages with less than 1,000 components. Click **Refresh Components** if the package has new or changed components and wait for the list to refresh.

Click **View Dependencies** to review a list of components that rely on other components, permissions, or preferences within the package. An entity might 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 56. 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 its versions.

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. Orgs entered in the **Push Upgrade Exclusion List** are omitted from the upgrade. The orgs can still install the upgrade when you publish the new version.

Click the Version Number of any listed uploads to manage that upload. For more information, see Managing Versions on page 285.

Note: **Push Upgrades** is available for patches and major upgrades. Registered ISV partners can request Push Major Upgrade functionality by logging a case in the Partner Community.

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 can no longer download or install this package. However, existing installations continue to work. • **Undeprecate**: Enables a previously deprecated package version to be installed again.
`Status`	The status of the package. The possible statuses are: • Released: The package is Managed - Released.

Attribute	Description
	• 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, first create a patch development organization. See Creating and Uploading Patches on page 276. Patch development organizations permit developers to change existing components without causing incompatibilities between existing subscriber installations. Click **New** to create a 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 in to your patch development organization. • **Reset**: Emails a new temporary password for your patch development organization.
`Administrator Username`	The login associated with the patch organization.
`Patching Major Release`	The package version number that you are patching.

Installing a Package

During the development and testing cycle, you might need to periodically install and uninstall packages before you install the next beta. Follow these steps to install a package.

Pre-Installation

1. In a browser, type in the installation URL you received when you uploaded the package.
2. Enter your username and password for the Salesforce organization in which you want to install the package, and then click **Log In**.
3. If the package is password-protected, enter the password you received from the publisher.

Default Installation

Click **Install**. You'll see a message that describes the progress and a confirmation message after the installation is complete.

Custom Installation

Follow these steps if you need to modify the default settings, as an administrator.

1. Choose one or more of these options, as appropriate.
   - Click **View Components**. You'll see an overlay with a list of components in the package. For managed packages, the screen also contains a list of connected apps (trusted applications that are granted access to a user's Salesforce data after the user and the application are verified). Review the list to confirm that the components and any connected apps shown are acceptable, and then close the overlay.

     Note: Some package items, such as validation rules, record types, or custom settings might not appear in the Package Components list but are included in the package and installed with the other items. If there are no items in the Package Components list, the package might contain only minor changes.

   - If the package contains a remote site setting, you must approve access to websites outside of Salesforce. The dialog box lists all the websites that the package communicates with. We recommend that a website uses SSL (secure sockets layer) for transmitting data. After you verify that the websites are safe, select **Yes, grant access to these third-party websites** and click **Continue**, or click **Cancel** to cancel the installation of the package.

     Warning: By installing remote site settings, you're allowing the package to transmit data to and from a third-party website. Before using the package, contact the publisher to understand what data is transmitted and how it's used. If you have an internal security contact, ask the contact to review the application so that you understand its impact before use.

   - Click **API Access**. You'll see an overlay with a list of the API access settings that package components have been granted. Review the settings to verify they're acceptable, and then close the overlay to return to the installer screen.
   - In Enterprise, Performance, Unlimited, and Developer Editions, choose one of the following security options.

     Note: Depending on the type of installation, you might not see this option. For example, in Group and Professional Editions, or if the package doesn't contain a custom object, Salesforce skips this option, which gives all users full access.

     Install for Admins Only

     Specifies the following settings on the installing administrator's profile and any profile with the "Customize Application" permission.

     - Object permissions—"Read," "Create," "Edit," "Delete," "View All," and "Modify All" enabled
     - Field-level security—set to visible and editable for all fields
     - Apex classes—enabled
     - Visualforce pages—enabled
     - App settings—enabled
     - Tab settings—determined by the package creator
     - Page layout settings—determined by the package creator
     - Record Type settings—determined by the package creator

     After installation, if you have Enterprise, Performance, Unlimited, or Developer Edition, set the appropriate user and object permissions on custom profiles as needed.

     Install for All Users

     Specifies the following settings on all internal custom profiles.

- Object permissions—"Read," "Create," "Edit," and "Delete" enabled
- Field-level security—set to visible and editable for all fields
- Apex classes—enabled
- Visualforce pages—enabled
- App settings—enabled
- Tab settings—determined by the package creator
- Page layout settings—determined by the package creator
- Record Type settings—copied from admin profile

Note: The Customer Portal User, Customer Portal Manager, High Volume Customer Portal, Authenticated Website, Partner User, and standard profiles receive no access.

Install for Specific Profiles...

Enables you to choose the usage access for all custom profiles in your organization. You can set each profile to have full access or no access for the new package and all its components.

- Full Access—Specifies the following settings for each profile.
  - Object permissions—"Read," "Create," "Edit," "Delete," "View All," and "Modify All" enabled
  - Field-level security—set to visible and editable for all fields
  - Apex classes—enabled
  - Visualforce pages—enabled
  - App settings—enabled
  - Tab settings—determined by the package creator
  - Page layout settings—determined by the package creator
  - Record Type settings—determined by the package creator
- No Access—Specifies the same settings as Full Access, *except* all object permissions are disabled.

You might see other options if the publisher has included settings for custom profiles. You can incorporate the settings of the publisher's custom profiles into your profiles without affecting your settings. Choose the name of the profile settings in the drop-down list next to the profile that you need to apply them to. The current settings in that profile remain intact.

Alternatively, click **Set All** next to an access level to give this setting to all user profiles.

2. Click **Install**. You'll see a message that describes the progress and a confirmation message after the installation is complete.

Post-Installation Steps

If the package includes post-installation instructions, they're displayed after the installation is completed. Review and follow the instructions provided. In addition, before you deploy the package to your users, make any necessary changes for your implementation. Depending on the contents of the package, you might need to perform some of the following customization steps.

- If the package includes permission sets, assign the included permission sets to your users who need them. In managed packages, you can't make changes to permission sets that are included in the package, but subsequent upgrades happen automatically. If you clone a permission set that comes with a managed package or create your own, you can make changes to the permission set, but subsequent upgrades won't affect it.
- If you're re-installing a package and need to re-import the package data by using the export file that you received after uninstalling, see Importing Package Data.
- If you installed a managed package, click **Manage Licenses** to assign licenses to users.

Note: You can't assign licenses in Lightning Experience. If you need to assign a license, switch to Salesforce Classic.

- Configure components in the package as required. For more information, see Configuring Installed Packages.

Component Availability After Deployment

Many components have an **Is Deployed** attribute that controls whether they are available for end users. After installation, all components are immediately available if they were available in the developer's organization.

For tips on customizing the installed package and components, see Configuring Installed Packages. Installed packages are available to users in your organization with the appropriate permissions and page layout settings.

Uninstalling a Package

To remove a package:

1. From Setup, enter `Installed Packages` in the Quick Find box, then select **Installed Packages**.
2. Click **Uninstall** next to the package that you want to remove.
3. Select **Yes, I want to uninstall...** and click **Uninstall**.
4. After an uninstall, Salesforce automatically creates an export file containing the package data, associated notes, and any attachments. When the uninstall is complete, Salesforce sends an email containing a link to the export file to the user performing the uninstall. The export file and related notes and attachments are listed below the list of installed packages. We recommend storing the file elsewhere because it's available for only two days after the uninstall completes, then it's deleted from the server.

   Tip: If you reinstall the package later and want to reimport the package data, see Importing Package Data.

When you uninstall packages, consider the following:

- If you're uninstalling a package that includes a custom object, all components on that custom object are also deleted. Deleted items include custom fields, validation rules, s-controls, custom buttons and links, workflow rules, and approval processes.
- You can't uninstall a package whenever a component not included in the uninstall references any component in the package. For example:
  - When an installed package includes any component on a standard object that another component references, Salesforce prevents you from uninstalling the package. An example is a package that includes a custom user field with a workflow rule that gets triggered when the value of that field is a specific value. Uninstalling the package would prevent your workflow from working.
  - When you have installed two unrelated packages that each include a custom object and one custom object component references a component in the other, you can't uninstall the package. An example is if you install an expense report app that includes a custom user field and create a validation rule on another installed custom object that references that custom user field. However, uninstalling the expense report app prevents the validation rule from working.
  - When an installed folder contains components you added after installation, Salesforce prevents you from uninstalling the package.
  - When an installed letterhead is used for an email template you added after installation, Salesforce prevents you from uninstalling the package.
- You can't uninstall a package that removes all active business and person account record types. Activate at least one other business or person account record type, and try again.
- You can't uninstall a package if a background job is updating a field added by the package, such as an update to a roll-up summary field. Wait until the background job finishes, and try again.

- Uninstall export files contain custom app data for your package, excluding some components, such as documents and formula field values.
- For some package types, you can also uninstall them with the Salesforce command-line interface (CLI).

Installing Managed Packages Using the API

You can install, upgrade, and uninstall managed packages using the API, instead of the user interface. Automating these repeated tasks can help you can work more efficiently and to speed up application development.

To install, upgrade, or uninstall a package, use the standard Metadata API `deploy()` call with the `InstalledPackage` metadata type. The following operations are supported.

- Deploying an `InstalledPackage` installs the package in the deploying organization.
- Deploying a newer version of a currently installed package upgrades the package.
- Deploying an `InstalledPackage` using a manifest called `destructiveChanges.xml`, instead of `package.xml`, uninstalls it from the organization.

Note: You can't deploy a package along with other metadata types. Hence, `InstalledPackage` must be the only metadata type specified in the manifest file.

The following is a typical project manifest (`package.xml`) for installing a package. The manifest must not contain a `fullName` or `namespacePrefix` element.

```
<?xml version="1.0" encoding="UTF-8"?>
  <Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
      <members>*</members>
       <name>InstalledPackage</name>
     </types>
    <version>28.0</version>
  </Package>
```

The package is specified in a file called ***MyNamespace***`.installedPackage`, where ***MyNamespace*** is the namespace prefix of the package. The file must be in a directory called `installedPackages`, and its contents must have this format.

```
<?xml version="1.0" encoding="UTF-8"?>
  <InstalledPackage xmlns="http://soap.sforce.com/2006/04/metadata">
    <versionNumber>1.0</versionNumber>
    <password>optional_password</password>
  </InstalledPackage>
```

InstalledPackage in API version 43.0 and later must include the `activateRSS` field set to either of these values.

true
Keep the isActive state of any Remote Site Settings(RSS) or Content Security Policies(CSP) in the package.

false
Override the isActive state of any RSS or CSP in the package and set it to `false`.

The default value is `false`.

Note: Regardless of what `activateRSS` is set to, a retrieve of `InstalledPackage` always returns `<activateRSS xsi:nil="true"/>`. Therefore, before you deploy a package, inspect the information you have retrieved from `InstalledPackage` and set `activateRSS` to the desired value.

To uninstall a package, deploy this `destructiveChanges.xml` manifest file in addition to the `package.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
  <Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
      <members>MyNamespace</members>
      <name>InstalledPackage</name>
     </types>
  </Package>
```

Retrieving an `InstalledPackage`, using the `retrieve()` call creates an XML representation of the package installed in an organization. If the installed package has a password, the password isn't retrieved. Deploying the retrieved file in a different organization installs the package in that organization.

For more information on the `deploy()` and `retrieve()` commands, see the *Metadata API Developer's Guide*.

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:

- 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.

EDITIONS

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

Available in: **Developer** Edition

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.

How does a Post Install Script Work?

Example of a Post Install Script

Specifying a Post Install Script

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.

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.

How does an Uninstall Script Work?

Example of an Uninstall Script

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.

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.

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

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

CHAPTER 6 Pass the AppExchange Security Review

In this chapter ...

- Security Review
- Security Review Steps
- Security Review Wizard
- Submit a Client or Mobile App for Security Review
- Submit an Extension Package for Security Review
- Security Review Resources
- Security Review FAQ

At Salesforce, nothing is more important than the security and success of our customers. To distribute a solution on AppExchange, it must pass our comprehensive security review. Learn how to design a solution security strategy and prepare for the security review.

Security Review

The security review ensures that the app or component you publish on the AppExchange meets industry best security standards. For the latest information on the security review, visit: http://p.force.com/security.

The AppExchange security review:

- Assures customers that your app or component works securely with Salesforce.
- Helps you deliver apps and components that span multiple systems and meet the needs of AppExchange customers.
- Allows Salesforce to facilitate open relationships between customers, developers, and providers by providing a secure ecosystem.

The scope of the security review depends on the type of offering.

Type	Description	Scope of Review
Lightning Platform	Offerings where the data, logic, and user interface is built entirely on the Lightning Platform platform.	• Automated code scan • Manual code review and black-box testing • Client-side components (Flash, JavaScript) • Integrations and web services
Client and Mobile Apps	Offerings that run outside of the Salesforce environment. It treats the Lightning Platform platform as a data source, using the development model of the tool and platform for which it's designed. Examples include iPhone apps and Microsoft Outlook connectors.	• Manual hands-on testing • Integrations and web services • Architecture review and web server testing
Web Apps	Offerings that run in a third-party hosted environment and integrate with Salesforce, leveraging the Lightning Platform Web-services API. The data, logic, and user interface can be stored outside of Lightning Platform.	• Automated testing and manual black-box testing • Client-side components (Flash, JavaScript) • Integrations and web services • Architecture review and web server testing

Security Review Steps

Before you can submit a product for security review, Salesforce must approve the product's business plan. Then have your product reviewed for security compliance.

1. Prepare for the security review.
   - Read the security guidelines.
   - Review the preparation tips found at http://p.force.com/security.
   - Review the free resources listed on our Secure Cloud Development site.
   - Review the Requirements Checklist.
   - Review the OWASP Top Ten Checklist.
   - Run a free self-service source code analysis against code developed on the Lightning Platform platform.
   - Run a web-application scan against your external web application that is integrated with Lightning Platform.

- Manually test your offering to ensure that it meets review requirements not found by tools. For details, see the OWASP Testing Guide.
- Fix any issues found during testing.

In general, be as thorough as you can in your testing. During the development phase of your app or component, run the code scanner several times to avoid fixing issues at the end. If you have questions, schedule office hours with the security review team at: http://security.force.com/security/contact/ohours.

2. Initiate the security review.

   Note: Before you initiate the security review, configure a test environment that Salesforce can use to test your offering. For information about setting up a test environment, see Required Testing Information for the ISV Security Review.

   a. Log in to the Partner Community.
   b. Open the security review wizard.
      - If your offering is a managed package, launch the wizard as follows.
        a. On the Publishing page, click the **Packages** tab.
        b. Find the offering that you want to submit and click **Start Review**.
      - If your have an API-only offering, launch the wizard as follows.
        a. On the Publishing page, click the **Listings** tab.
        b. Find the offering that you want to submit and click it to open the AppExchange publishing console.
        c. Click the **App** tab and choose the API-only option.
        d. Click **Start Security Review**.
   c. Follow the steps outlined in the security review wizard, which guides you through the rest of the submission.
   d. If this is a paid offering, pay the annual listing fee and the one-time security review fee. If your app or component is free, these fees are waived.

   After you submit your package, the security review team runs tests to identify potential vulnerabilities. If necessary, they will contact you to discuss their findings. The review team performs both application and network security testing and sends you the results.

3. Review the results. There are three possible outcomes.
   - **Approved**—You can list your app or component on the AppExchange and distribute it to customers immediately.
   - **Provisional Pass**—The security review team identified low- or medium-risk issues that can be addressed easily and do not pose significant risks. You can create a public listing for your offering on the AppExchange and distribute it to customers. If you don't fix the issues identified within the specified time period, your app or component is removed from the AppExchange.
   - **Not Approved**—The security review team identified high-risk issues during the testing phase. You can't list your offering on the AppExchange or distribute it to customers until all issues have been addressed and your offering has been reviewed again. If the app or component is already listed on the AppExchange, you must address the issues within 60 days. Because the security review is a black-box, time-limited process, we can't list every instance in which a particular issue occurred. Interpret these findings as representative examples of the types of issues you must fix across the offering.

Key Steps to Follow after Passing the Security Review

When you have passed the security review, you can:

- List your offering publicly on the AppExchange and distribute it to customers.
- Request API access for your offering in Professional and Group Editions.

Important: Salesforce reserves the right to conduct periodic reviews of all offerings. If we find that the app or component doesn't meet our security requirements, we notify you and provide time to remedy the issue. In extreme cases, we pull the AppExchange listing from public viewing. In this case, you must cease distribution of your offering.

Security Review Wizard

Use the online security review wizard to submit information about your offering to Salesforce for testing. The wizard is comprehensive, so give yourself plenty of time to respond to the questions. Be as thorough as you can, and remember that your responses are saved as you go—you can always return later to complete the process. The more information you provide, the faster we can test and approve your app or component.

The wizard consists of a series of screens that guide you through the information required.

1. Preparation

View tips and links to resources to help you prepare for the security review.

2. General Information

Add information for the person at your company who we can contact with security-related questions.

3. Policies and Certifications

Attach your company's information security policy and certifications that you've earned. For example, ISO 27001.

4. Components

List the technologies used by your app or component. You can select relevant items in a checklist based on the type. Examples include:

- Force.com — Apex, Visualforce, API, SSO
- Web app — frameworks and languages (Java, .NET, Rails, SSO, Heroku, and so on)
- Client app — desktop app, and browser plug-in
- Mobile app — iOS, Android, BlackBerry, Windows

5. Test Environments

Provide fully configured environments for testing, including login credentials, install links, and sample data.

- Force.com — usernames and passwords for all user levels (admin, end user, and so on) in a test organization
- Web app — URLs, usernames, and passwords for all user levels, API keys, SSO, and OAuth/SAML settings
- Client app — install URLs, configuration data and instructions, required license files, associated sample data, config guides, credentials
- Mobile app — separate install link for each type of mobile app

6. Reports

Upload reports from your testing.

- Force.com — Security Code Scanner report

  Note: Makes sure that the code scanner results are clean. If you're aware of issues in the scanner report that are false positives, provide the details.

- Web — Web App Scanner report
- Other — any other reports or documentation that you want to provide

7. Review Details

Review a summary of the information you've provided to verify that your submission is correct and complete. If there's something you'd like to change, you can modify it.

8. Payment

Pay for the security review using Recurly. Salesforce saves your payment information, so you only need to provide it once. If your app or component is free, no payment is required.

Important: If you've already paid the security review fee for your offering, you aren't charged again. However, you're still asked to confirm the payment information every time you run the security review wizard.

Submit a Client or Mobile App for Security Review

Most of the client app requirements, and some of the web app requirements, apply to mobile apps. Here are typical scenarios:

- The mobile app has a component that sits on the customer's organization. The component is a managed package and follows the security review process for a packaged app.
- The client app only uses APIs for communicating with Salesforce. In this case, follow the process for an API-only app for security review.

For testing, we ask that you provision us an app for all the platforms that you plan to distribute. We can accept a test flight or an ad hoc deployment for iOS. For other platforms, we can accept the app in a file (APK, COR, and so on). Similar to a composite app, if there are callouts to anything other than Salesforce, we ask for a web application scanner report. We accept Zed Attack Proxy (ZAP) and Burp reports. If the mobile app has a web component, even if it's optional, Salesforce requires a web application scanner report.

Submit an Extension Package for Security Review

ISVs create extension packages when they want to provide add-on features to their apps. The extension packages also help when ISVs want to support Salesforce editions like PE and GE with their app. Another use case is creating a "bridge" package that enables the ISV's app to work with another app.

All packages, whether base or extension, require a security review. Follow the same process for review of an extension package as for a base package.

Some extension packages are small, for example, a few links or buttons to call base package components. Regardless of the size of the extension package, follow the same process. The only difference is that the review process is faster for smaller packages.

The process for submitting an extension package for the security review is similar to the process for the base package.

1. Upload your extension package (be sure it is managed-release like your base package). Upload the extension package from an organization other than the org for base package.
2. In your AppExchange listing, link the organization where the extension package was created. The extension package appears in the list of packages under your listings.
3. Initiate the security review. Make sure that your test account includes both the base and extension packages.

It's important that the Salesforce security team reviews every extension package. Even small packages can introduce vulnerabilities to the platform. Follow the same process of doing a self-scan of the code before submitting for a review. If the extension package has components that interface with an external application, run a web application scan, such as Zed Attack Proxy (ZAP) or Burp. Submit the corresponding results.

Security Review Resources

These resources can help you prepare for the security review.

- Security Review Process

- Security Review Requirements Checklist
- Secure Cloud Development Resources
- Secure Coding Guidelines
- Open Web Application Security Project (OWASP)
- OWASP Top 10 Issues
- OWASP Testing Guide
- OWASP Secure Coding Guide
- OWASP Secure Coding Practices Quick Reference

Security Review FAQ

This section contains a list of frequently asked questions about the security review.

- Is an AppExchange security review required?
- What happens during a security review?
- Why do I need to have a security review?
- How long does the security review take? How often is it required?
- Is there a fee for the security review?
- Why do I have to test my offering before the review if the security team is going to test it anyway?
- What are the typical reasons why I would not pass the security review?
- Can I submit my offering before it's complete to get the security review process done early?
- If I have any "No" responses in the security review wizard, or no formal and detailed documentation, do I fail the review?
- Why does the review team need to test the X or Y part of my offering?
- Do I have to fix all the issues that the security review team reported?
- Why can't the review team send me every instance of every finding for my review?
- What happens after I pass the security review?
- What happens if my offering isn't approved?
- What's the difference between Approved, Provisional Pass, and Not Approved?
- When I update my offering, do I need to pay the security review fee again to have it reviewed?
- When I create a managed package to upgrade my offering, do I need to pay the security review fee again?
- Why perform periodic security reviews?
- How do reviewed solutions work with PE and GE organizations?

Is an AppExchange security review required?

Yes. Every app and component that is publicly listed on the AppExchange must pass a security review.

What happens during a security review?

The security review process includes two kinds of assessments:

- Qualitative—question and answer round-to-review policies and procedures
- Quantitative—network and application penetration tests (see Scope)

Why do I need to have a security review?

We want the AppExchange to be a trusted on-demand application ecosystem. The security review helps build this culture of trust by ensuring that our offerings adhere to security standards and best practices. This review lends credibility to the AppExchange and, in turn, helps increase customer adoption.

How long does the security review take? How often is it required?

The review process takes about 6–8 weeks, provided that:

- Your documentation is complete and accurate.
- The test environment is complete, fully configured, and includes all necessary information.
- You've met all the requirements.
- You're within the agreement guidelines.

Periodically, we review apps or components again. The timeline varies depending on the security risk of the offering, but it's usually between six months and two years since the last review. Around the expiry date, Salesforce contacts you to arrange another review.

Is there a fee for the security review?

If your offering is paid, Salesforce charges a fee for the initial review, and a small annual fee for subsequent reviews. If your offering is free, these fees are waived. Pricing information is available in the Partner Community.

Why do I have to test my offering before the review if the security team is going to test it anyway?

By testing your offering before you submit, you're more likely to pass the review on the first round. Applicants who don't test beforehand rarely pass and must resubmit after addressing security concerns. Resubmitting significantly delays the publishing process.

What are the typical reasons why I would not pass the security review?

In no particular order, here's a list of the top reasons for not being approved. For more information, see OWASP Top 10.

- Injection (SQL, XML, and so on)
- Cross-site scripting
- Broken authentication and session management
- Insecure direct object references
- Cross-site request forgery
- Security misconfiguration
- Insecure cryptographic storage
- Failure to restrict URL access
- Insufficient transport layer protection
- Unvalidated redirects and forwards

Can I submit my offering before it's complete to get the security review process done early?

No. It's important that the offering you submit is complete and the version that you intend to distribute. If the offering is not what you intend to distribute or is incomplete, we can't properly identify issues. We will need to review again, which delays your offering.

If I have any "No" responses in the security review wizard, or no formal and detailed documentation, do I fail the review?

No. Although the questions ask for formal and detailed documentation, we understand that, depending on the size and maturity of your company, this might not be possible. We designed the wizard to cover a broad spectrum of companies, from small to large. Smaller companies can provide us with a checklist instead of formal documentation of their security policies. If you have to answer "No" to some questions, use the comments box to explain. We understand that the scoring isn't black and white, and we factor in elements like company size and maturity to guide our decision-making.

Why does the review team need to test the X or Y part of my offering?

Our approach is to test all parts of the offering to ensure that our mutual customers and their data are not put at risk. This includes external web applications or services that are required or optional, client/mobile applications that are required or optional, and all Apex and Visualforce (packaged or unpackaged) that is included in the offering. If you're not sure if you should include part of your offering, include it anyway. The review team will not test parts of the offering that we determine are not in scope, but if a required part is not included, your review will be delayed.

Do I have to fix all the issues that the security review team reported?

Yes. Unless otherwise noted on the test reports, you're required to fix all classes of issues that the review team detected across the entire app or component.

Why can't the review team send me every instance of every finding for my review?

The security review is a black-box, time-limited review, and it would be impractical to provide that information given the visibility and time the review team has for each review. The findings should be interpreted as examples and all issues in the provided categories should be fixed across the offering.

What happens after I pass the security review?

After you pass the security review, you can list your offering publicly on the AppExchange and distribute it to customers. If your app uses the SOAP or REST APIs, you can request an API token. You can only associate an approved package version with your public listing. Every package version associated with a public listing must be submitted for security review.

If your offering passed the security review within the last year, the new version is auto-approved, and its status changes to Passed. The status change can take up to 24 hours.

What happens if my offering isn't approved?

If your offering isn't approved, you must address the security review team's findings and submit the app or component for a follow-up review. When your offering is approved, you can publish on the AppExchange and distribute it to customers.

What's the difference between Approved, Provisional Pass, and Not Approved?

If you're granted Approved or Provisional Pass on your security review, it means we didn't find any high-risk vulnerabilities in your offering. A Provisional Pass indicates that we likely found medium-risk vulnerabilities, and we work out a mutually acceptable timeline for you to remedy these items. In the meantime, you can list your app or component on the AppExchange and distribute it to customers. For a review that wasn't approved, you must remedy the issues found before you can list your offering on the AppExchange or get access to the API token.

When I update my offering, do I need to pay the security review fee again to have it reviewed?

No. When you upload a new package version to the AppExchange and associate it with your listing, we run a source code analysis to identify security vulnerabilities. This scan is included in the annual listing fee, so there is no extra charge. However, you are asked to confirm your payment information when you run the security review wizard.

When I create a managed package to upgrade my offering, do I need to pay the security review fee again?

No. If you developed the new version with a package that we've previously approved, it's automatically approved when you submit it for review. However, you are asked to confirm your payment information when you run the security review wizard.

Why perform periodic security reviews?

We require periodic security reviews for all apps and components published on the AppExchange. These reviews usually occur six months to two years after the initial approval, depending on the risk of the offering. Periodic reviews ensure that upgraded apps and components continue to meet security best practices and aren't subject to the latest security vulnerabilities. We also update our review process so that we're aligned with industry requirements.

How do reviewed solutions work with PE and GE organizations?

Eligible partners can apply for access to the API in the Partner Community, which enables approved offerings to make API calls. If you're an eligible partner and have passed the security review, request API access by logging a case in the Partner Community. See Use of ClientID for code examples.

CHAPTER 7 Publish Your Solution on AppExchange

In this chapter ...

- What Is the AppExchange?
- Business Plans for AppExchange Listings
- Publish on the AppExchange
- Email Notifications
- AppExchange Checkout
- Checkout Management App
- Work with AppExchange Leads
- Analytics Reports for Publishers
- Update the Package in an AppExchange Listing
- AppExchange FAQ

You turned an idea into a solution and are ready to get it in front of customers. To publish your solution on AppExchange, use the AppExchange publishing console. The publishing console is where you create a listing for your solution, connect orgs, manage license settings, and view analytics for your published listings.

What Is the AppExchange?

The AppExchange is an online marketplace for Salesforce apps, components, and consulting services. If you're a Lightning Platform developer or consultant, the AppExchange is the gateway for connecting customers to your business solution. If you're a Salesforce administrator or user, visit the AppExchange to find tools and talent to unleash your company's productivity.

How Does the AppExchange Work?

An AppExchange listing is your primary marketing tool for promoting your app or component. In the listing, you can describe your solution, pricing, support, and other details so that customers can determine if your offering is right for them. You also have a chance to upload videos, white papers, and other content to help customers understand what you are delivering. Based on the information you provide, an AppExchange curator categorizes the listing into one or more business areas, like sales, marketing, or analytics.

After you've created a provider profile and uploaded your package, you can create a listing. You can create only one listing per app or component. This approach has several advantages. As the provider, it's easier to maintain and upgrade your offering over its lifecycle. Having one listing also helps you achieve a higher ranking, because the metrics that the AppExchange uses to rank apps and components, like page views, aren't diluted across multiple listings. Customers benefit, too, because your offering is easier to find, all your reviews are in one place, and there aren't several similar listings to cause confusion.

Who Can Use the AppExchange?

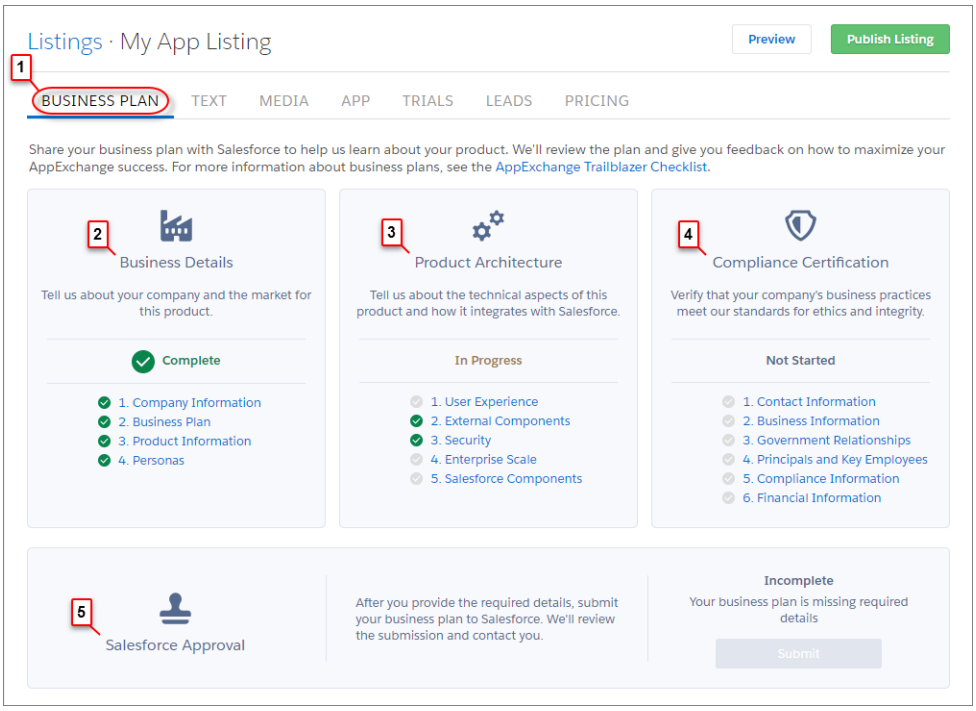
Anyone can browse listings and test-drive apps or components. You need the "Download Packages" permission to install apps or components. To create a package and upload it to the Partner Community, you must have "Create Packages" and "Upload Packages" permissions. To create and publish a listing, you must have the "Manage Listings" permission.

Business Plans for AppExchange Listings

To publish a listing on AppExchange, we ask you to provide a business plan for your app, Lightning component, or other product. A business plan tells us about your company and the product you're building. It helps us verify that you meet our standards for ethics and integrity. Salesforce must approve your business plan before you can submit your product for security review. If you recently joined the AppExchange Partner Program, you can sign your partnership agreement after we approve your business plan.

You can create and manage the business plan for your listing on the Business Plan tab (1) in the AppExchange publishing console. A business plan has these sections.

Section	Purpose
Business Details (2)	Share information about your company, the market for your product, and its target users. We use this information to understand how your company fits into the Salesforce product ecosystem.
Product Architecture (3)	Share technical information about your product, such as how it stores credentials, passwords, and other sensitive data. We use this information to verify that your product follows Salesforce best practices for architecture and design.
Compliance Certification (4)	Share information about your company's business practices. We use this information to verify that you meet our standards for ethics and integrity. Note: We ask you to provide compliance information only for paid AppExchange listings.



After you finish your business plan, submit it for review. We contact you to discuss your partnership and then either approve the plan or return it to you with comments. If your plan is returned, you can resubmit it when you've addressed our comments. To check the status of your plan, go the Salesforce Approval section (5).

After your business plan is approved, we contact you with instructions for signing your partnership agreement. To check the status of the agreement, go to the AppExchange Partner Agreement section (6). If you're an existing partner, you've already signed an agreement, so this section doesn't display.

Publish on the AppExchange

To publish an app, component, or consulting service on the AppExchange, follow these high-level steps.

1. If your listing is an app or component, connect your packaging organization to the AppExchange.
2. Create a provider profile.
3. Review tips for creating a listing that excites and engages customers.
4. Create the listing.
5. If your listing is an app or component, submit a business plan for review.
6. If your listing is an app or component, submit the package for security review.

7. After your app is approved, publish the listing on the AppExchange.
8. Review the analytics to see how your listing is performing.

Connect a Packaging Organization to the AppExchange

To publish a listing on the AppExchange, first connect the packaging organization, which is the organization that contains your packaged app or component.

1. Log in to the Partner Community.
2. On the Publishing page, click the **Organizations** tab.
3. Click **Connect Organization**.
4. Enter the login credentials for the organization that contains the package you want to list.
5. Click **Submit**.

If the AppExchange finds any packages, they appear on the **Packages** tab on the Publishing page. From the **Packages** tab, create an app or component listing, or begin the security review.

Create or Edit Your Provider Profile

A polished, accurate provider profile is a key part of establishing customer trust in your app, component, or consulting service. On your profile, you can share a mission statement and tell customers where you're located, how many employees you have, and so on. People browsing listings see this information on the Provider tab.

To create or edit your profile, open the Publishing page in the Partner Community, and then go to the **Company Info** tab.

Create or Edit an AppExchange Listing

Market your app, component, or consulting service with an AppExchange listing. Create a listing or edit an existing one in the AppExchange publishing console, which guides you through the process.

To create or edit a listing, open the Publishing page in the Partner Community, and click the **Listings** tab.

Here are the tabs you navigate when creating or editing a listing.

Tab	What you do:	Available on these listings types:
Business Plan	• Add a business plan for your offering • If you're a standard ISVforce partner, sign your Salesforce partnership agreement	App, Component
Text	• Describe your offering • Provide contact information so that customers and Salesforce can get in touch	App, Component, Consulting Service
Media	• Add branding • Upload images, videos, and other resources to help customers understand your offering	App, Component, Consulting Service

Tab	What you do:	Available on these listings types:
App	• Upload the package that contains your app (or the link to your app if you're only using the Salesforce API)	App
Component	• Upload the package that contains your component	Component
Trials	• Set up a test drive or free trial so that customers can see your offering in action	App, Component
Leads	• Choose how Salesforce collects leads when customers interact with the listing	App, Component, Consulting Service
Pricing	• Choose whether your offering is free or paid and provide pricing information	App, Component
Service Offering	• Choose listing categories, like services offered and industry focus	Consulting Service

Add a Business Plan to an AppExchange Listing

Before submitting your product for security review, add a business plan to your AppExchange listing. The business plan includes details about your company and its operations, your product architecture, and compliance information. To add a business plan, go to your product listing in the AppExchange publishing console.

If your listing is paid, provide pricing details before you add a business plan. Otherwise, you can't provide compliance information. If your listing is free, we don't collect compliance information.

USER PERMISSIONS

To edit AppExchange listings:
- Manage Listings

1. Log in to the Partner Community.
2. Click **Publishing**.
3. On the Listings tab, click a listing tile.
4. On the Business Plan tab, provide details about your company and product architecture.
5. If your listing is paid, provide compliance information.

   Note: If you're an existing partner with another published paid listing, we already have your compliance information, so this section is marked as complete.

6. Click **Submit for Approval**.

After you submit your business plan, we contact you to discuss your partnership. You can check the plan's approval status on the Business Plan tab.

Make Your AppExchange Listing Effective

A great app, component, or consulting service deserves a listing to match. We gathered feedback from customers and Salesforce marketing experts to provide a list of tips to make your listing stand out.

Tell Customers, Then Show Them

An effective listing combines concise, customer-oriented writing with compelling visuals. As you craft your listing, keep the following tips in mind.

- **Emphasize a use case**—When customers read your listing, they want to understand the problem you're solving, whether they're part of the target audience, and what makes your offering different. As you explain your solution, put things in terms the customer cares about. For example, if your component helps support reps resolve cases 10% faster, say so.
- **Add screenshots, videos, and demos**—Customers are more likely to interact with listings that have visuals. Most people like to at least see how something works before making a purchase.
- **Make the listing easy to read**—Like you, the typical AppExchange customer is busy. Help customers understand what's important by making your listing easy to read. Keep sentences short and use formatting, like bullets, to draw attention to key points. If you've added screenshots or a video, use zooming and annotations to highlight features.

Aim for Clean and Simple Design

An effective listing tends to have a clean and simple design. When making design decisions, keep the following tips in mind.

- **Find a design reference**—Before you create a logo, banner, or other graphic, find a design that you like and think about what it does well. For example, does it use a visually pleasing font? Keep these ideas in mind as you begin designing.
- **Preview before publishing**—The AppExchange lets you preview your listing before publishing, and you can see exactly how your offering will appear to customers. Put yourself in the customer's shoes and ask, "If I saw this listing, would I feel comfortable buying this app or component?"

For more tips, see *Partner Logo and Branding Usage Guidelines* in the Education section of the Partner Community.

Choose an Installation Option

The easier it is for people to install your offering, the more likely it is they will become paying customers. Choose the installation option that gives customers the best experience.

Option	When to choose this:
`Directly from the AppExchange`	If your offering is packaged, this option provides the simplest installation experience. It allows people to install your offering into their Salesforce sandbox or production environment through the AppExchange installation sequence without assistance from you. This option is required for components and recommended for apps.
`From your website`	If your app is a downloadable client or needs additional information to be installed, this option is the best. After users click **Get It Now** on your listing and agree to the terms and conditions, they are directed to your website to complete the installation process. Make sure that you've provided clear download instructions and performed the required setup or configuration.
`They should contact us to install it`	If your installation or selection process requires your assistance, you must choose this option. After agreeing to terms and conditions, the customer is told that you'll be in touch shortly to help with installation. Make sure that your company has the resources to assist potential customers.

Register Your Package and Choose License Settings

If you register a package and set up the License Management App (LMA), you receive a license record each time a customer installs your app or component. Licenses let you track who is using your app or component and for how long.

Note: Before you register a package, make sure that:

- Your app or component is in a managed package.
- You have installed the LMA. In most cases, the LMA is installed in your partner business organization.

1. Log in to the Partner Community.
2. On the Publishing page, click the **Packages** tab.
3. Click **Manage Licenses** next to the package that you want to register.
4. Click **Register**. Enter the login credentials for the organization where the LMA is installed. Usually, the organization is your partner business organization.
5. Select whether your default license is Free Trial or Active.
6. If you selected a free-trial license, enter the length of the trial, up to 90 days.
7. Enter the number of seats associated with your default license, or select **License is site-wide** to offer the license to all users in the installer's organization.
8. Click **Save**.

Submit Your Listing for Security Review

To distribute your app or component to customers, or provide a free trial, you must pass the AppExchange security review. This review ensures that your offering is safe for customers to install.

Important: You are contractually required to keep security review information current. For example, if you upgrade your component to use a new web service, you must edit the information in your security review submission.

Submit an App or Component Listing for Review

1. Log in to the Partner Community.
2. On the Publishing page, click the **Packages** tab.
3. Find the package that you want to submit, and then click **Start Review**.
4. Follow the steps in the security review wizard to complete the submission.

Within two days, a member of the partner operations team will contact you with next steps.

Submit a Trial Template for Review

Before you submit a trial template for security review, make sure that the packages that it contains have passed the security review.

1. Log in to the Partner Community.
2. On the Publishing page, click the **Trial Templates** tab.
3. Find the trial template that you want to submit, and then click **Start Review**.
4. Follow the on-screen prompts to create a profile in your trial organization that the security review team can use for testing. If you already have a profile, click **Skip to the next step**.

5. Enter the login credentials for the test profile.
6. Click **Submit Security Review**.

Email Notifications

Installation Notification Emails

Salesforce emails your subscribers 30 days after they install your app or component. The email thanks subscribers and encourages them to share their experiences with others by writing a review. We only send emails when:

- The subscriber has a valid email address.
- The subscriber hasn't already received a notification.
- The subscriber hasn't yet posted a review.

Review Notification Emails

When subscribers post reviews and comments on your listings, Salesforce emails parties who are likely to be interested. The notification that subscribers receive depends on their role in the conversation (provider, author, or commenter).

Type of Email Notification	Sent to	Details
New Review on Your Listing	You, the provider	Sent whenever someone posts a review of your listing.
New Comment on Your Review	The review author	Sent only if someone other than the review author comments on the review and if the author has opted to receive email notifications on their profile. If the author replies to the notification, the reply is posted as a new comment on the review.
Also Commented on the Review	The people who commented on the review	Sent to people who have commented on a review, are not the review author or the author of this comment, and have opted to receive email notifications on their profiles. At most, one email notification is sent to each commenter for each new comment. If the person replies to the notification, the reply is posted as a new comment on the review.
New Comment on the Review of Your Listing	You, the provider	Sent whenever someone writes a new comment on a review of your listing.

AppExchange Checkout

AppExchange Checkout is the easy way to manage payments for your AppExchange product. Powered by Stripe, Checkout lets customers pay with a credit card or bank payment, so buying is simple. Checkout also integrates automatically with the License Management App and Checkout Management App. That means product licenses stay up-to-date and subscription insights are available in just a few clicks.

Note: AppExchange Checkout is available in English only to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, visit https://partners.salesforce.com.

You're Looking For:	AppExchange Checkout:
A simple but flexible way to manage purchases and trials	Is built on Stripe, an industry leader in online payments. Stripe handles the entire purchase process, from collecting billing details to processing payments. For each listing, you choose a pricing model and the payment methods that you accept. If you want to give your product an extra boost in the marketplace, Stripe also supports trials and coupons.
Easy licensing	Integrates with the License Management App, so you don't have to update license settings manually when customers upgrade, renew, or cancel.
Insights about your subscribers	Integrates with the Checkout Management App, which lets you see revenue, subscription status, and other product data in customizable dashboards.

The fee to use AppExchange Checkout is 15%. We don't charge setup fees, monthly service charges, or card storage fees. There's also no minimum revenue share.

Tip: Stop by Trailhead and earn the AppExchange Checkout badge before you get started with Checkout. You'll learn how Checkout helps you manage online payments, monitor sales for your solutions, and have an opportunity to test your knowledge.

Pricing Models in AppExchange Checkout

AppExchange Checkout supports one-time payments and subscriptions. For either pricing model, you can charge customers on a per user or per company basis. You can also combine multiple pricing models on a listing.

Payment Methods in AppExchange Checkout

AppExchange Checkout supports credit card payments and debit transfers from US or European bank accounts. You can accept credit cards, bank account transfers, or both payment methods on a listing where Checkout is enabled.

AppExchange Checkout Considerations

Keep these considerations in mind when using AppExchange Checkout.

Get Started with AppExchange Checkout

To use AppExchange Checkout, first create a Stripe account and configure subscription plans for your product. After Stripe is set up, enable Checkout on your AppExchange listing and choose a pricing model and the payment methods that you accept. If your customers live in regions that charge a Value Added Tax (VAT), configure VAT for Checkout transactions.

AppExchange Checkout FAQ

Answers to common questions about AppExchange Checkout.

Pricing Models in AppExchange Checkout

AppExchange Checkout supports one-time payments and subscriptions. For either pricing model, you can charge customers on a per user or per company basis. You can also combine multiple pricing models on a listing.

Here are the pricing models that you can offer.

Pricing Model	Pricing Options	Customer Is Billed:	You Set Up Payment Plans In:
One-time payment	• Per User • Per Company	Once, at the time of purchase	Your Stripe account

Pricing Model	Pricing Options	Customer Is Billed:	You Set Up Payment Plans In:
Subscription	• Per User • Per Company	On a recurring basis, either monthly or annually	The Publishing Console on the Partner Community

A single pricing model may not work well for all customers, so we recommend offering a few options on your listing.

Payment Methods in AppExchange Checkout

AppExchange Checkout supports credit card payments and debit transfers from US or European bank accounts. You can accept credit cards, bank account transfers, or both payment methods on a listing where Checkout is enabled.

Payment Method	You Can Receive Payments From:	Notes
Credit cards	Visa, MasterCard, American Express, JCB, Discover, or Diners Club credit cards.	Payments are processed immediately.
US bank accounts	Checking, savings, or money market accounts based in the United States. Payments are processed using the Automated Clearing House (ACH) network.	• Payments can take up to 5 days to process. • Your subscription plan must be in US dollars (USD). You can manage the currency of subscription plans in Stripe. • Customers must pay with a business bank account. Checkout doesn't support ACH payments from personal bank accounts. • Customers must have a US billing address.
European bank accounts	Checking, savings, or money market accounts based in the European Union. Payments are processed using the Single Euro Payment Area (SEPA) framework.	• Payments are processed immediately. • Your subscription plan must be in euros (EUR). You can manage the currency of subscription plans in Stripe. • Customers must have an EU billing address.

Your company's location in Stripe determines the type of bank transfers that you can accept. To accept ACH payments, your company must be based in the United States. To accept SEPA payments, your company must be based in the European Union. You can't accept both ACH and SEPA payments.

AppExchange Checkout Considerations

Keep these considerations in mind when using AppExchange Checkout.

- You must distribute your product as a managed package.
- You can't use Checkout with OEM apps.
- You can't accept payments from every country. For a list of supported countries, visit https://stripe.com/global. If a country isn't listed, sign up to be notified when Stripe is available there.

Get Started with AppExchange Checkout

To use AppExchange Checkout, first create a Stripe account and configure subscription plans for your product. After Stripe is set up, enable Checkout on your AppExchange listing and choose a pricing model and the payment methods that you accept. If your customers live in regions that charge a Value Added Tax (VAT), configure VAT for Checkout transactions.

1. Set Up Stripe for AppExchange Checkout

   Before you enable AppExchange Checkout on a listing, create and configure an account with our payment partner, Stripe. After Stripe is set up, payments on your Checkout-enabled listings are deposited in the bank account that you connect to Stripe.

2. Create a Subscription Plan for AppExchange Checkout

   To charge customers on a recurring basis, create a subscription plan in Stripe. A subscription plan lets you determine how much your customers are charged, how often, and other payment details. After you connect your Stripe account to AppExchange, your subscription plans appear in the Publishing Console.

3. Activate Bank Payments for AppExchange Checkout

   To let customers pay for your product with bank account transfers, request this payment method in Stripe. After Stripe reviews and approves your request, you're eligible to receive bank payments. Depending on your location, you can accept payments through the Automated Clearing House (ACH) network or the Single Euro Payment Area (SEPA) framework.

4. Enable AppExchange Checkout on a Listing

   After you create a Stripe account and configure subscription plans for your product, you can enable AppExchange Checkout on a listing. When you enable Checkout, you choose a pricing model and the payment methods that you accept.

5. Collect VAT for Transactions

   If your customers live in regions that charge a Value Added Tax (VAT), you can include VAT in AppExchange Checkout transactions. After you enable this option, VAT is applied to invoices in Stripe. You're responsible for VAT registration, maintaining required data, and distributing the taxes that you collect.

6. Convert a Free Listing to Paid, with Checkout

   If you have a free listing that passed security review, you can convert the listing to accept payments using AppExchange Checkout. First, enable Checkout for the listing in the Publishing Console. Then, log a case in the Partner Community to pay the security review fee. The fee is waived for free listings, but is collected when you charge for your product.

Set Up Stripe for AppExchange Checkout

Before you enable AppExchange Checkout on a listing, create and configure an account with our payment partner, Stripe. After Stripe is set up, payments on your Checkout-enabled listings are deposited in the bank account that you connect to Stripe.

USER PERMISSIONS

To manage AppExchange listings:

- Manage Listings

Have the following information available when you create your Stripe account.

- A short description of your business, such as the products you sell
- Basic information about your business, like its physical address
- Your login information for an external identity provider, such as Google, Facebook, or LinkedIn
- Account and routing numbers for the bank account where you want to receive funds

After you've gathered these details, you're ready to set up Stripe.

1. Log in to the Partner Community.
2. Click **Publishing**.
3. Create a listing or edit an existing one.

4. On the Pricing tab, select **Paid, Using Checkout**.
5. Click **Set Up Stripe**.
6. Complete your Stripe account application and submit. If you already have a Stripe account, you can sign in instead.

After you create your account, you can manage it on the Stripe website. To learn more about Stripe accounts, go to https://stripe.com/docs/dashboard.

Create a Subscription Plan for AppExchange Checkout

To charge customers on a recurring basis, create a subscription plan in Stripe. A subscription plan lets you determine how much your customers are charged, how often, and other payment details. After you connect your Stripe account to AppExchange, your subscription plans appear in the Publishing Console.

Tip: To charge a one-time payment for your product, you don't need to create a subscription plan. Instead, create the one-time payment on the Pricing tab of the Publishing Console.

To create a subscription plan, go to https://dashboard.stripe.com/plans (login required). Have the following information available.

- Unique name and ID for the plan
- Monthly or annual billing (Stripe offers other billing intervals, but Checkout doesn't support them)
- Amount to charge customers and in which currency

  Note: To let customers pay with US bank accounts, you must use US dollars (USD) for the plan. To let customers pay with European bank accounts, you must use euros (EUR) for the plan.

- Trial period length, if any

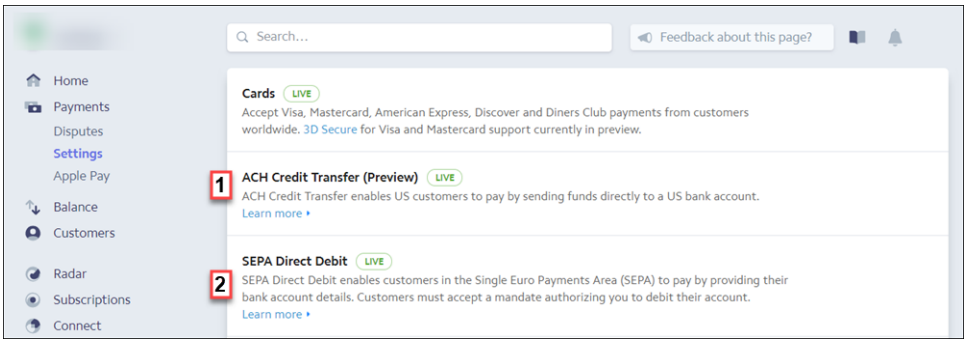
To learn more about subscription plans, go to https://stripe.com/docs/subscriptions/guide.

Activate Bank Payments for AppExchange Checkout

To let customers pay for your product with bank account transfers, request this payment method in Stripe. After Stripe reviews and approves your request, you're eligible to receive bank payments. Depending on your location, you can accept payments through the Automated Clearing House (ACH) network or the Single Euro Payment Area (SEPA) framework.

Note: Your company's location in Stripe determines the type of bank transfers that you can accept. To accept ACH payments, your company must be based in the United States. To accept SEPA payments, your company must be based in the European Union. You can't accept both ACH and SEPA payments.

1. Go to the Stripe website.
2. Log in to your Stripe account.
3. In the Stripe dashboard, click **Payments**.
4. Click **Settings**.
5. Request ACH Credit Transfer (1) or SEPA Direct Debit (2) for your account.



Your activation request is sent to Stripe for processing. You receive an email when your request is approved.

6. If you requested ACH Credit Transfer, verify that the activation succeeded.
   a. Go to the Stripe website again.
   b. Log in to your Stripe account.
   c. Go to Stripe's ACH Guide.
   d. Click **Enable ACH**. If you don't see an option to enable ACH, ACH Credit Transfer is already active for your account.

Enable AppExchange Checkout on a Listing

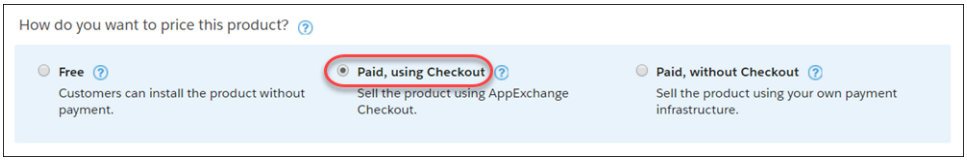
After you create a Stripe account and configure subscription plans for your product, you can enable AppExchange Checkout on a listing. When you enable Checkout, you choose a pricing model and the payment methods that you accept.

USER PERMISSIONS

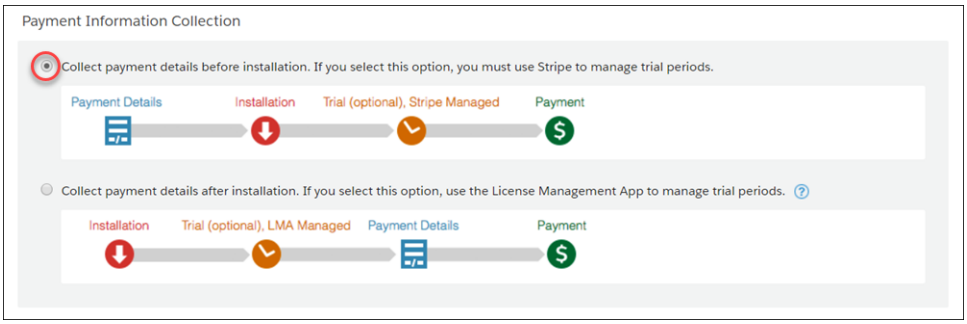
To manage AppExchange listings:

- Manage Listings

1. Log in to the Partner Community.
2. Click **Publishing**.
3. Create a listing or edit an existing one.
4. On the Pricing tab, select **Paid, using Checkout**.



5. Select when to collect payment details from the customer.

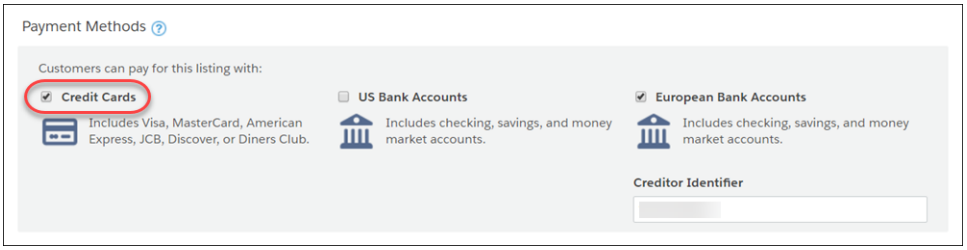


6. Select pricing models (1) and adjust the display order (2).

Tip: You can offer several pricing models on a listing. For example, you can combine a one-time payment with one or more subscription plans.

Pricing Model	Steps
Subscription	**a.** Select one of the subscription plans that you created in Stripe. **b.** Select company or per-user pricing.
One-time payment	**a.** Click **Add One Time Price Option**. **b.** Provide a name for the plan. **c.** Select a price and currency. **d.** Select company or per-user pricing.

7. Select the payment methods that you accept.



Note: Your company's location in Stripe determines the type of bank payments that you can accept. Before you enable bank payments, verify that you activated ACH Credit Transfer (for US Bank Accounts) or SEPA Direct Debit (for European Bank Accounts) in Stripe. You receive an email from Stripe when your account is ready to receive bank payments.

8. Click **Save**.

Collect VAT for Transactions

If your customers live in regions that charge a Value Added Tax (VAT), you can include VAT in AppExchange Checkout transactions. After you enable this option, VAT is applied to invoices in Stripe. You're responsible for VAT registration, maintaining required data, and distributing the taxes that you collect.

USER PERMISSIONS

To manage AppExchange listings:
- Manage Listings

1. Log in to the Partner Community.
2. Click **Publishing**.
3. Click **Company Info**.
4. Select the option to collect VAT on purchases.

   Note: VAT isn't supported for one-time purchases.

5. Enter a VAT number and country. The country you enter must match the country specified in Stripe.
6. Click **Save**.

If you manage Checkout data with the Checkout Management App, you can use the app to view information for VAT reporting.

Convert a Free Listing to Paid, with Checkout

If you have a free listing that passed security review, you can convert the listing to accept payments using AppExchange Checkout. First, enable Checkout for the listing in the Publishing Console. Then, log a case in the Partner Community to pay the security review fee. The fee is waived for free listings, but is collected when you charge for your product.

USER PERMISSIONS

To manage AppExchange listings:
- Manage Listings

1. Log in to the Partner Community.
2. Enable Checkout for the listing.
   a. Click **Publishing**.
   b. Click **Listings**.
   c. Find the listing you want to update and then click the tile.
   d. On the Pricing tab, select **Paid, using Checkout** and configure pricing details.
   e. Click **Save**.
3. Log a support case in the Partner Community.
   a. Click **Support**.
   b. Click **New Case**.
   c. For topic, choose Security Review and click **Create a Case**.
   d. Explain that you're converting a free listing to paid using AppExchange Checkout and provide your listing's URL.

e. Click **Submit Case**.

Salesforce will contact you to arrange for payment of the security review fee.

AppExchange Checkout FAQ

Answers to common questions about AppExchange Checkout.

- Does AppExchange Checkout replace the License Management App?
- How does AppExchange Checkout affect lead management and Trialforce?
- Do I collect customer payment details before or after installation?
- Does AppExchange Checkout support purchases in multiple currencies?
- Does revenue share apply to transactions that aren't processed with AppExchange Checkout?
- Do customers have to purchase my offering on the AppExchange or can I add them as customers in Stripe?
- Can customers switch payment plans on the AppExchange?
- If a credit card is declined, does a license become inactive?
- How does billing work when customers add or remove licenses during the billing period?
- If an admin installs an offering, can others in the company go to their AppExchange account to Buy Now?
- Does AppExchange Checkout Support Stripe's Tiered Pricing Feature?

Does AppExchange Checkout replace the License Management App?

No, AppExchange Checkout doesn't replace the License Management App (LMA). AppExchange Checkout works with the LMA to create and update license records and control subscription status for Checkout purchases. You can't directly edit the license records created by AppExchange Checkout. Instead, ask the customer to update subscription information on the My Account page in the AppExchange.

How does AppExchange Checkout affect lead management and Trialforce?

AppExchange Checkout doesn't affect how leads are sent or your Trialforce configuration. However, it does change how trial licenses are managed. When a customer signs up for a trial using AppExchange Checkout, the corresponding trial user is listed as Active in the License Management App.

Do I collect customer payment details before or after installation?

We recommend thinking about your target customers and the existing business processes at your company and then deciding. You can use this table to guide your thinking.

Approach	Best If Your Customers Want:	Best If Your Company:
Collect details *before* installation	A quick and easy purchase experience at the end of a trial	Is comfortable managing trials in Stripe
Collection details *after* installation	To get up and running with an offering as quickly as possible	Prefers managing trials from your partner business org using the License Management App

Does AppExchange Checkout support purchases in multiple currencies?

Yes. When you sign up for Stripe, you choose a default payment currency based on your country (for example, USD if you're in the United States). You can enable other currencies in your Stripe account settings. When customers purchase your offering, AppExchange Checkout charges them in your specified currency and deposits the money into your bank account.

Does revenue share apply to transactions that aren't processed with AppExchange Checkout?

Revenue share applies to Stripe transactions associated with the AppExchange. As an AppExchange Checkout partner, you agree that all purchases of your offering occur on the AppExchange and are subject to revenue sharing.

Do customers have to purchase my offering on the AppExchange or can I add them as customers in Stripe?

Customers have to purchase your offering on the AppExchange. If you add customers to plans using Stripe, the AppExchange can't associate them with your listing or provision licenses through the License Management App.

Can customers switch payment plans on the AppExchange?

No, customers must contact you. Then, you manually switch them to the new plan in Stripe.

If a credit card is declined, does a license become inactive?

You specify in your Stripe account settings what happens when a credit card is declined. You can retry the payment or deactivate the subscription. If you deactivate the subscription, the license becomes inactive.

How does billing work when customers add or remove licenses during the billing period?

If licenses are added or removed during the billing period, AppExchange Checkout charges the customer a prorated amount.

If an admin installs an offering, can others in the company go to their AppExchange account to Buy Now?

Any user who has permissions to install offerings can perform Buy Now actions, provided the user also has the "Manage Billing" permission. This permission is the same one required to view the My Account page or make purchases using AppExchange Checkout inside the Salesforce app.

Does AppExchange Checkout Support Stripe's Tiered Pricing Feature?

No. If you add pricing tiers to a payment plan in Stripe, Checkout can't import the plan to the Publishing Console. If you'd like to offer discounted pricing for a product that you sell, create a coupon in Stripe and provide it to customers.

Checkout Management App

The Checkout Management App (CMA) brings the power of Salesforce to AppExchange Checkout. A beautiful dashboard visually displays AppExchange Checkout data, so it's easy to see how your offerings are performing. Automated email notifications keep customers and team members in the loop whenever activity occurs on your offerings.

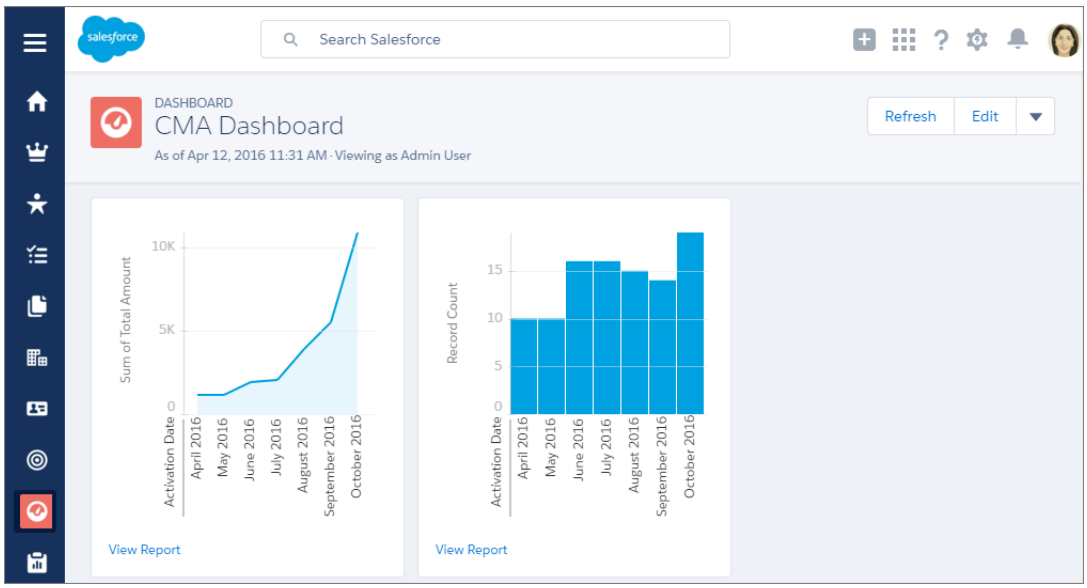
Note: The CMA is available in English and Japanese to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, visit https://partners.salesforce.com.

Start with the dashboard to get a big picture view of your AppExchange Checkout data.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

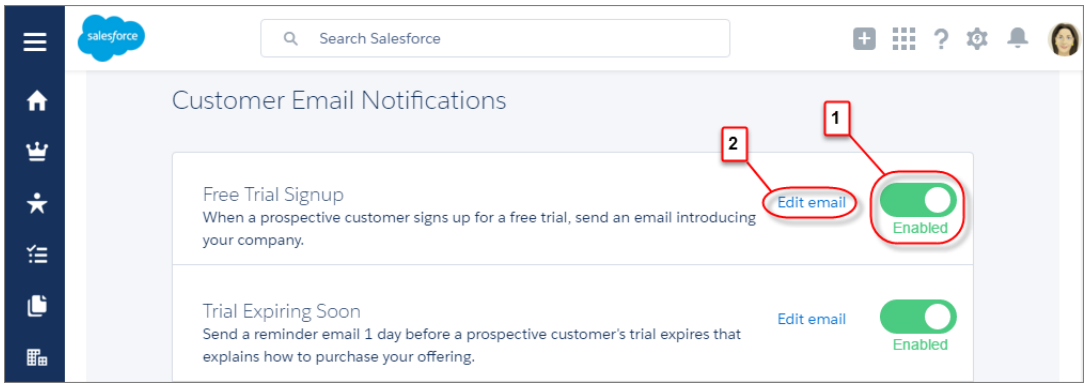


The dashboard is preconfigured to show:

- Revenue by month, so financial performance is always front and center
- New subscribers by month, so it's easy to see where growth is occurring
- Subscription plan by unit, so you know which configurations are popular with customers
- Subscription status by month, so you can stay on top of trials, purchases, and renewals

You can customize the dashboard using standard Salesforce tools. For a detailed look at your data, view individual customer, subscription plan, subscription, invoice, invoice item, and transaction records.

To save time communicating with stakeholders, the CMA can send email notifications for situations that you often encounter as a partner, like renewal notices. Enable email notifications as needed (1) and then customize them to reflect your company's identity (2). Not in the mood to customize anything? No worries—the templates provide friendly and informative default content.



Checkout Management App Best Practices

Follow these guidelines and best practices when you use the Checkout Management App (CMA).

Checkout Management App Objects

Subscription plan, subscription, invoice, invoice item, and transaction objects are the foundation of the Checkout Management App (CMA). To get the most out of the CMA, understand what these objects represent and how they relate to each other.

Get Started with the Checkout Management App

Install the Checkout Management App (CMA) into a Salesforce org, and then configure the app so that users get the right level of access to data. Enable email notifications to simplify communication with customers and team members. You can also customize the notification templates to meet your company's needs.

Sample Checkout Management App Customizations

The Checkout Management App (CMA) is a powerful tool out of the box, but gets even better when you customize it. These examples show how you can modify dashboards and email notifications to delight customers and team members.

Update Settings in the Checkout Management App

Control when customers and team members receive emails from the Checkout Management App (CMA). You can also change the Stripe account associated with the CMA and manually reimport your data into your Salesforce org. Only admin users can update settings in the CMA.

View Checkout Management App Logs

The Checkout Management App (CMA) creates logs when connecting to Stripe or syncing your data. If you experience issues with the CMA, view logs to help diagnose their cause.

Checkout Management App Best Practices

Follow these guidelines and best practices when you use the Checkout Management App (CMA).

- Install the CMA in a Salesforce org where the License Management App (LMA) is already installed. Usually, this is your partner business org. If the LMA isn't installed in your org, you can't install the CMA.
- Don't edit data in managed fields on the subscription plan, subscription, invoice, or transaction object records. The CMA syncs Stripe data in a one-way, read-only manner, so changes that you make aren't reflected in Stripe. To update subscription plan, subscription, invoice, invoice item, or transaction data, use the Stripe dashboard or API.
- Review and customize notification templates before enabling them. By adding your logo and tailoring template content to reflect your company's identity, you set yourself apart from other offerings on the AppExchange. Customizing takes only a couple of minutes and doesn't require any coding.

Checkout Management App Objects

Subscription plan, subscription, invoice, invoice item, and transaction objects are the foundation of the Checkout Management App (CMA). To get the most out of the CMA, understand what these objects represent and how they relate to each other.

The CMA pulls in data from AppExchange Checkout's payment partner, Stripe, to populate the subscription plan, subscription, invoice, invoice item, and transaction objects. Here's a high-level overview of these objects and how they fit together.

Object	Purpose	Relationships
Subscription plan (1)	Contains information about the pricing model of an offering. For example, site-wide or per user, billed monthly.	Parent object of: • Subscription
Subscription (2)	Contains information about the customer's history and usage of an offering. For example, when the subscription started.	Child object of: • Subscription plan Parent object of: • Invoice • Transaction
Invoice (3)	Contains billing and payment information for a subscription for a specific time period. For example, the total amount owed by the customer.	Child object of: • Subscription Sibling object of: • Transaction
Invoice item (4)	Contains information about a particular billing and payment event for a specific time period. For example, a one-time credit. Multiple invoice items can be associated with an invoice.	Child object of: • Invoice
Transaction (5)	Contains information about a customer payment attempt. For example, method of payment and whether it was successful.	Child object of: • Subscription Sibling object of: • Invoice

We haven't listed it in the table, but there's one more object to be aware of: customer. The customer object contains information about the subscriber and draws from the other objects in the CMA, including subscription, invoice, and transaction.

The CMA automatically syncs new data from Stripe, updating object records as necessary. Just remember: syncing is one way and read only, so changes that you make to object records aren't reflected in Stripe. To update subscription plan, subscription, invoice, invoice item, or transaction data, use the Stripe dashboard or API.

Get Started with the Checkout Management App

Install the Checkout Management App (CMA) into a Salesforce org, and then configure the app so that users get the right level of access to data. Enable email notifications to simplify communication with customers and team members. You can also customize the notification templates to meet your company's needs.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Install the Checkout Management App

Install the Checkout Management App (CMA) in the Salesforce org where you manage licenses, usually your Partner Business Org. The License Management App (LMA) is required to use the CMA, so make sure that you install the LMA in this org first.

Set Up the Checkout Management App

Use the Checkout Management App (CMA) setup tool to connect your Stripe account and import data into your Salesforce org. Then get familiar with your dashboard and choose when customers and team members receive email notifications from the CMA.

Assign Access to the Checkout Management App

Use permission sets to give team members the right level of access to the Checkout Management App (CMA). You can assign the CMA Standard User permission set or CMA Admin User permission set, depending on the features team members must access.

Modify a Notification Template in the Checkout Management App

The Checkout Management App (CMA) can send email notifications in response to trial installations, purchases, and other subscription changes. We created default notifications to get you started, but you can tailor templates to your company's needs.

Configure Logs in the Checkout Management App

The Checkout Management App (CMA) creates debug logs to help you troubleshoot issues. By default, all logs are saved, but you can configure the CMA to delete logs that you no longer need. Delete logs regularly to stay within the data storage limits for your Salesforce edition.

Install the Checkout Management App

Install the Checkout Management App (CMA) in the Salesforce org where you manage licenses, usually your Partner Business Org. The License Management App (LMA) is required to use the CMA, so make sure that you install the LMA in this org first.

Note: If you received a Partner Business Org when you joined the Partner Community, the CMA is preinstalled there. To check if the CMA is installed in your org, go to the App Launcher and look for the CMA in the list of available apps.

USER PERMISSIONS

To install packages:

- Download AppExchange Packages

1. If you haven't already, log in to the AppExchange using the credentials of the org where you want to install the CMA.
2. Go to the AppExchange listing for the CMA: https://appexchange.salesforce.com/listingDetail?listingId=a0N3A000000rMclUAE.
3. Click **Get It Now**.
4. Click **Install in production**.
5. Agree to the Terms & Conditions, and then click **Confirm and Install**.
6. Log in to the org where you want to install the CMA.
7. Review the package installation details, and then click **Continue**.
8. Approve access by third-party websites, and then click **Continue**.
9. Review the API access requirements for the package, and then click **Next**.
10. Grant access to package contents, and then click **Next**.

    Note: Salesforce recommends granting access to admins only and assigning access to other users as needed after the app is installed.

11. Click **Install**.
12. After the installation completes, go to the App Launcher and confirm that the CMA appears in the list of available apps.

Set Up the Checkout Management App

Use the Checkout Management App (CMA) setup tool to connect your Stripe account and import data into your Salesforce org. Then get familiar with your dashboard and choose when customers and team members receive email notifications from the CMA.

USER PERMISSIONS

To configure the Checkout Management App:
- CMA Admin User

Watch a Demo: Set Up the Checkout Management App

1. Log in to the org where the CMA is installed.
2. Open the App Launcher, and then click **Checkout Management App**.
3. Click **Checkout Setup**.
4. Connect your Stripe account.
   a. In the Connect Stripe Account section, click **Do It**.
   b. Click **Get API Key from Stripe**.
      The Stripe dashboard opens in a new tab.
   c. In the Stripe dashboard, copy your live secret API key.
   d. In the CMA, paste the key into `Live Secret API Key`, and then click **Connect Stripe Account**.
5. Set up data syncing by creating and configuring a site. After you set up data syncing, new Stripe data syncs to your org automatically.
   a. Click **Set Up Data Syncing**.
   b. Click **Register a Force.com Domain**, and then follow the setup instructions in the CMA.
   c. Click **Create a Force.com Site**, and then follow the setup instructions in the CMA.
   d. Click **Configure Site Access**, and then follow the setup instructions in the CMA.
   e. Click **Connect the Site to Stripe**, and then follow the setup instructions in the CMA.
6. Import your Stripe data. If you haven't sold an offering using AppExchange Checkout before, you don't have any Stripe data, so you can skip this step.
   a. Click **Import Existing Data**.
   b. Click **Import Data**.
      Importing Stripe data can take awhile depending on how much data you have. Don't use CMA reports or dashboards while data is being imported.
   c. After the import finishes, close the dialog to return to the setup wizard.
7. Configure email notifications.

   Tip: Before you enable a notification, review the default content we provide. That way, you know exactly what customers and team members receive, and you can tailor it to reflect your company's identity.

   a. In the Configure Notification Settings section, click **Do It**.
   b. Enable customer notifications as desired.
   c. To add the email addresses of team members, click **View/Edit**, and then click **Save**.
   d. Enable partner notifications as desired.
   e. Go back to the setup wizard.
8. Say hello to your dashboard.
   a. In the Meet Your Dashboard section, click **Do It**.

b. View the dashboards we've created for you, or go to Trailhead to learn how to customize dashboards.

You're all set! To update configuration details later, return to Checkout Setup.

Assign Access to the Checkout Management App

Use permission sets to give team members the right level of access to the Checkout Management App (CMA). You can assign the CMA Standard User permission set or CMA Admin User permission set, depending on the features team members must access.

Standard users have read-only access to the dashboard and object records and can't view or update notification settings. System Admins or users with the CMA Admin User permission set have full access to the dashboard, notifications, and objects, including the ability to edit objects. Assign the CMA Admin User permission set only to users who administer or manage the CMA.

USER PERMISSIONS

To assign a permissions set:
- Assign Permission Sets

1. Log in to the org where the CMA is installed.
2. From Setup, enter `Users` in the `Quick Find` box, and then click **Users**.
3. Select a user.
4. In the Permission Set Assignments related list, click **Edit Assignments**.
5. Select the CMA Standard User or CMA Admin User permission set, and then click **Add**.
6. Click **Save**.

Modify a Notification Template in the Checkout Management App

The Checkout Management App (CMA) can send email notifications in response to trial installations, purchases, and other subscription changes. We created default notifications to get you started, but you can tailor templates to your company's needs.

Notification templates in the CMA are based on Visualforce email templates. The templates support advanced customizations, like merge fields and formulas.

Note: Notification templates in the CMA also include custom components that affect email styling. You can't modify these components, but you can remove them.

USER PERMISSIONS

To enable, disable, or customize notifications:
- CMA Admin User

To create or change Visualforce email templates:
- Customize Application

1. Log in to the org where the CMA is installed.
2. Open the App Launcher, and then click **Checkout Management App**.
3. Click **Checkout Notification Settings**.
4. Find the template that you want to customize, and then select **Edit**.
5. Click **Edit Template** and modify as needed, and then click **Save**.

Configure Logs in the Checkout Management App

The Checkout Management App (CMA) creates debug logs to help you troubleshoot issues. By default, all logs are saved, but you can configure the CMA to delete logs that you no longer need. Delete logs regularly to stay within the data storage limits for your Salesforce edition.

USER PERMISSIONS

To manage, create, edit, and delete custom settings:
- Customize Application

To save changes to Apex classes and triggers:
- Author Apex

1. Log in to the org where the CMA is installed.
2. Configure how long to save CMA logs.
   a. From Setup, enter `Custom Settings` in the `Quick Find` box, and then click **Custom Settings**.
   b. For CMALogSettings, click **Manage**.
   c. Click **New**.
   d. Enter a name. For example, `CMA Log Settings`.
   e. For CMALogLifeSpan, enter how many days to save logs. For example, enter `30` to save all logs created in the past 30 days.

   Note: To change how long CMA logs are saved, edit the value configured in this step. Don't add more values to CMALogSettings.

3. Schedule an Apex job to delete old CMA logs.
   a. From Setup, enter `Apex Classes` in the `Quick Find` box, and then click **Apex Classes**.
   b. Click **Schedule Apex**.
   c. Configure the job as follows.

Field	Value
`Job Name`	CMA Log Cleanup
`Apex Class`	ScheduledDeleteCMALogs Note: Namespace prefix: `sfcma`
`Frequency`	Specify a weekly or monthly interval—we recommend running the job at least once per week
`Start Date`	Today's date
`End Date`	A future date—we recommend specifying a date that's at least several years in the future
Preferred Start Time	Any value—we recommend choosing a time when your org is not under heavy load

   d. Click **Save**.

Sample Checkout Management App Customizations

The Checkout Management App (CMA) is a powerful tool out of the box, but gets even better when you customize it. These examples show how you can modify dashboards and email notifications to delight customers and team members.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Use an Organization-Wide Address on a Notification

By default, notifications sent by the Checkout Management App (CMA) include a generic email address in the From field. But what if you want to include contact information for a specific team at your company, like support or billing? You can specify an organization-wide address on a notification so that customer replies are directed to the right people at your company.

Include a Link in a Notification

When a customer installs your offering, you often want to provide information that doesn't fit in the notification, such as setup documentation. You can point customers to this information by including links in a Checkout Management App (CMA) notification.

Customize a Report to Show Annual Revenue for an Offering

If the Checkout Management App (CMA) dashboard doesn't show what you need out of the box, try modifying a report. This example steps you through how to display annual revenue for an offering instead of monthly revenue across all offerings.

Use an Organization-Wide Address on a Notification

By default, notifications sent by the Checkout Management App (CMA) include a generic email address in the From field. But what if you want to include contact information for a specific team at your company, like support or billing? You can specify an organization-wide address on a notification so that customer replies are directed to the right people at your company.

Suppose that your company's refund inquiries are fielded by a billing specialist whose email address is `billing@example.com`. Let's step through how to add this email address to the Refund Notification template so that customers know who to contact if they have questions.

USER PERMISSIONS

To enable, disable, or customize notifications:
- CMA Admin User

To configure organization-wide addresses:
- Modify All Data

1. Log in to the org where the CMA is installed.
2. Create an organization-wide email address.
   a. From Setup, enter *`Organization-Wide Addresses`* in the `Quick Find` box, and then click **Organization-Wide Addresses**.
   b. Click **Add**.
   c. For the display name, enter a word or phrase that users who receive the email see as the sender. For this example, enter *`Billing Support`*.
   d. Enter an email address. For this example, enter *`billing@example.com`*.
   e. Choose which profiles can use the address. For this example, enable the address for all profiles.
   f. Click **Save**.
3. Add the organization-wide email address to the notification template.
   a. From Setup, enter *`Email Alerts`* in the `Quick Find` box, and then click **Email Alerts**.
   b. Find the notification template that you want to update, and then click **Edit**. For this example, choose the Refund Customer Notification template.
   c. For `From Email Address`, choose an organization-wide email address. For this example, choose *`"Billing Support" <billing@example.com>`*.

4. Click **Save**.

Include a Link in a Notification

When a customer installs your offering, you often want to provide information that doesn't fit in the notification, such as setup documentation. You can point customers to this information by including links in a Checkout Management App (CMA) notification.

Suppose that you sell a product that requires configuration after it's installed. To help customers get off on the right foot, direct them to a page on your website that offers configuration tips. Let's step through how to add a link to the Free Trial Signup template.

USER PERMISSIONS

To enable, disable, or customize notifications:
- CMA Admin User

To create or change Visualforce email templates:
- Customize Application

1. Log in to the org where the CMA is installed.
2. Open the App Launcher, and then click **Checkout Management App**.
3. Click **Checkout Notification Settings**.
4. Find the template that you want to use, and then click **Edit**. For this example, choose the Free Trial Signup template.
5. Click **Edit Template**.
6. Modify the email template to include the `<apex:outputLink>` component, which lets you point to an external URL. For this example, add this component after the last sentence in the message body.

```
<apex:outputLink value="https://example.com/getstarted" target="_blank">Check out our
website for configuration tips.</apex:outputLink>
```

Note: The `target` attribute is set to blank, which opens the URL in a new page.

7. Click **Save**.

Customize a Report to Show Annual Revenue for an Offering

If the Checkout Management App (CMA) dashboard doesn't show what you need out of the box, try modifying a report. This example steps you through how to display annual revenue for an offering instead of monthly revenue across all offerings.

USER PERMISSIONS

To customize CMA reports:
- CMA Admin User

To create, edit, and delete reports:
- Create and Customize Reports

  AND

  Report Builder

1. Log in to the org where the CMA is installed.
2. Open the App Launcher, and then click **Checkout Management App**.
3. Click **Dashboards**, and then click **CMA Dashboard**.
4. For the Revenue Per Month chart, click **View Report**.
5. From the Edit drop-down list, select **Clone**.
6. Specify field values as follows, and then click **Create**.

Field Name	Value
`Name`	*`Revenue Per Year`* To keep your dashboard organized, include the name of your offering. For example, Revenue Per Year (Sample App).
`Folder`	CMA Reports

7. Click **Edit**.
8. Add a filter to display revenue for a specific offering.
   a. From the Add drop-down list, select **Field Filter**.
   b. Enter filter criteria. To display revenue only for listings named Sample App, create the filter `Listing Name equals Sample App`.
   c. Click **OK**.
9. In the Preview section, from the Activation Date drop-down list, select **Group Dates By** > **Calendar Year**.
   Now the report is set up to show annual revenue instead of revenue by month.
10. Click **Save**, and then click **Run Report**.

Update Settings in the Checkout Management App

Control when customers and team members receive emails from the Checkout Management App (CMA). You can also change the Stripe account associated with the CMA and manually reimport your data into your Salesforce org. Only admin users can update settings in the CMA.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Change Notification Settings in the Checkout Management App

You can enable or disable individual Checkout Management App (CMA) email notifications depending on your customers' and team members' needs.

Change the Stripe Account Associated with the Checkout Management App

If you start managing subscriptions from another Stripe account, update your account settings in the Checkout Management App (CMA) to keep Stripe data in sync.

Reimport Stripe Data into the Checkout Management App

The Checkout Management App (CMA) automatically pulls new Stripe data into your org, so usually you don't need to import anything manually. However, if data in the CMA is missing or incorrect, you can manually reimport Stripe data.

Change Notification Settings in the Checkout Management App

You can enable or disable individual Checkout Management App (CMA) email notifications depending on your customers' and team members' needs.

USER PERMISSIONS

To enable, disable, or customize notifications:
- CMA Admin User

1. Log in to the org where the CMA is installed.
2. Open the App Launcher, and then click **Checkout Management App**.
3. Click **Notification Settings**.
4. Enable or disable a customer or partner notification.

Change the Stripe Account Associated with the Checkout Management App

If you start managing subscriptions from another Stripe account, update your account settings in the Checkout Management App (CMA) to keep Stripe data in sync.

USER PERMISSIONS

To configure the Checkout Management App:
- CMA Admin User

1. Log in to the org where the CMA is installed.
2. Open the App Launcher, and then click **Checkout Management App**.
3. Click **Checkout Setup**.

4. In the Connect Stripe Account section, click **Change**.
5. Note: If you change or disconnect the current Stripe account, existing Stripe data in your org remains.

   To associate a new Stripe account, click **Change Stripe Account**, and then enter a new live secret API key.

Reimport Stripe Data into the Checkout Management App

The Checkout Management App (CMA) automatically pulls new Stripe data into your org, so usually you don't need to import anything manually. However, if data in the CMA is missing or incorrect, you can manually reimport Stripe data.

Warning: The reimport process overwrites existing Stripe data in your org. Changes you've made to existing data are lost. Report and dashboard customizations and notification settings aren't affected.

USER PERMISSIONS

To configure the Checkout Management App:
- CMA Admin User

1. Log in to the org where the CMA is installed.
2. Open the App Launcher, and then click **Checkout Management App**.
3. Click **Checkout Setup**.
4. In the Import Existing Data section, select **Re-import Data**.
5. Confirm that you want to overwrite the existing Stripe data, and then click **Yes, Reimport Data**.

View Checkout Management App Logs

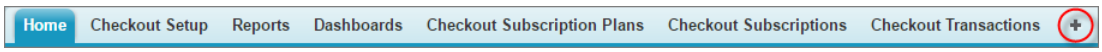
The Checkout Management App (CMA) creates logs when connecting to Stripe or syncing your data. If you experience issues with the CMA, view logs to help diagnose their cause.

USER PERMISSIONS

To manage apps:
- Customize Application

To view CMA logs:
- CMA Admin User

1. Log in to the org where the CMA is installed.
2. To view CMA logs in Lightning Experience:
   a. Open the App Launcher, and click **Other Items**.
   b. Click **Checkout Logs**.
3. To view CMA logs in Salesforce Classic:
   a. Open the App Launcher, and click **Checkout Management App**.
   b. Click the plus icon (+) next to the main tabs.



   c. Click **Checkout Logs**.

Work with AppExchange Leads

When a customer interacts with your listing, the AppExchange collects contact information that can be delivered to you as a lead.

You can collect leads when a customer:

- Installs your app or component

- Takes a test drive
- Watches a demo or video
- Signs up for a free trial
- Clicks the **Learn More** link

Before you enable lead collection on a listing, make sure that you've set up Web-to-Lead in the organization where you want to receive leads.

AppExchange Leads FAQ

This section contains a list of frequently asked questions about AppExchange Leads.

- How do I receive leads from the AppExchange?
- Can I choose to receive leads for one listing but not others?
- How will I know this lead came from the AppExchange?
- Can Salesforce de-duplicate leads before they are sent to me?
- What do the lead source codes from my listing mean?
- How can I see more information in the lead record?
- What's the difference between the leads and license records my listing generates?
- What happens to my listing if I chose not to receive AppExchange leads?

How do I receive leads from the AppExchange?

To receive leads from the AppExchange, edit your AppExchange listing and click the **Leads** tab. Specify the actions you want to receive leads for and to which Salesforce organization to send the leads. This organization must have Web-to-Lead enabled and be a standard Salesforce organization, not a Developer Edition organization. We recommend using your partner business org so that you can manage leads and licenses from a single, convenient location.

Can I choose to receive leads for one listing but not others?

Yes, you can enable lead collection for one listing but not others. Lead collection is enabled on a per listing basis. If you don't want to collect leads for a particular listing, don't choose that option in the publishing console.

How will I know this lead came from the AppExchange?

The lead source codes give you information about how the lead was created and can help you determine how to proceed. The lead source code always takes the form of `SFDC-XX|Listing Name` or `SFDC-dup-XX|Listing Name`. If the source code contains `-dup-`, AppExchange already sent you a lead for this user, listing, or action within the last 180 days. The `XX` identifies the action the user took to generate the lead.

Can Salesforce de-duplicate leads before they are sent to me?

A majority of the partners we polled requested that we send them all the leads. We designate duplicate AppExchange leads as follows: `SFDC-dup-XX|Listing Name`.

If the source code contains `-dup-`, then AppExchange already sent you a lead for this user, listing, or action within the last 180 days. For example, `SFDC-dup-DM|VerticalResponse for AppExchange` indicates a duplicate lead from a user who clicked to view the demo video on the VerticalResponse for AppExchange listing.

What do the lead source codes from my listing mean?

The lead source codes give you information about how the lead was created and can help you determine how to proceed. The lead source code always takes the form of `SFDC-XX|Listing Name` or `SFDC-dup-XX|Listing Name`. If the source code contains `-dup-`, AppExchange already sent you a lead for this user, listing, or action within the last 180 days. The `XX` identifies the action the user took to generate the lead.

Here's a table of the action codes and what they mean.

Lead Source Code	Description
IN	The user started the install process for your app or component by clicking **Get It Now** on your listing, agreeing to the terms and conditions, and clicking the install button on the confirmation page. The user might not have completed the installation or might have uninstalled the app or component. Use the License Management Application (LMA) to track who has the package installed.
DM	The user watched some or all of your demo.
LM	The user clicked **Learn More** on your listing. Note: Listings that previously had Learn More buttons now have Get It Now buttons and receive lead source codes with IN actions.
TS	The user clicked **Get It Now** on your listing and chose to start a new 30-day free trial of Salesforce and your app or component. These users might be preexisting Salesforce customers.
TD	The user took your test drive by clicking **Test Drive** on your listing.

How can I see more information in the lead record?

At this time, providers can't modify the lead form that customers are asked to fill out when they view a demo, access the test drive, install an app, or click to Learn More. Please indicate any improvements you wish to see on the IdeaExchange.

What's the difference between the leads and license records my listing generates?

Leads and license records are generated by specific actions that a customer takes on your listing. If you've set up Web-to-Lead and enabled lead collection on your listing, leads are generated when a customer does any of the following: views a video or demo, clicks **Learn More**, takes a test drive, or installs your app or component. By contrast, license records are generated only when a customer installs your app or component. You must also have the License Management App enabled in your partner business org to receive licenses.

What happens to my listing if I chose not to receive AppExchange leads?

If you don't select a scenario for collecting leads, customers aren't prompted to fill out the lead sign-up form, and no leads are sent. Regardless of the lead settings, customers can still view the demo, take a test drive, click the **Learn More** link, and install your app or component.

Analytics Reports for Publishers

AppExchange analytics reports are powerful visual tools for understanding how your app, component, or consulting partner listing is performing. These reports provide metrics related to the web traffic, number of installations, and other user activities over time. By looking at the reports, you can quickly gain insights about the aspects of your listing that resonate with customers and which areas need refinement.

- To access a report for your listing, open the Publishing page in the Partner Community, and then click the **Analytics** tab.

Report Types

For app and component listings, the available reports are:

- Installs (Get It Now)
- Leads
- Resources & Promotions
- Test Drives, Demos & Screenshots
- Web Analytics

For a consulting partner listing, the available reports are:

- Leads
- Learn Mores, Videos & Screenshots
- Web Analytics

Report Attributes

All the reports share these common attributes.

Listing Name
The title of the listing shown at the top of every report.

Back to Publishing Home link
Returns you to the Publishing Home page.

Show Menu
Allows you to choose from one of the available reports. The reports are sorted alphabetically.

Date Range Menu
Allows you to choose the date range. Last 30 Days is selected by default.

Metrics

Report	Metrics
Installs (Get It Now)	Get it Now, Installs, Click-to-Install Ratio

Report	Metrics
Leads	Unique Leads, Duplicate Leads, Total Leads
Resources & Promotions	Case Studies, Data Sheets, Promotions, Customer Testimonials, Webinars, Customization Guides, Whitepapers
Test Drives, Demos & Screenshots	Test Drives, Demos, Screenshots
Learn Mores, Videos & Screenshots	Learn Mores, Videos, Screenshots
Web Analytics	Page Views, SEO Searches, Visits, Internal Searches, Unique Visitors

Line Graph

Shows one or more lines for each metric you've selected for display. Select the checkboxes beneath the graph for the metrics you want to see. By default, all metrics are included in the graph. The reports show metrics over time grouped by created date. When you click the graph, the date and selected metrics for that date display. Next to each metric, the number of items in the metric over the selected date range displays regardless of whether you have chosen to include the graph of that metric.

Table

Each report includes a table. The first column on all reports is the Date, and the rest of the columns correspond to the metrics associated with the report. The table shows 30 rows at a time. Click **Next** to see more data. By default, the table is sorted by date from oldest to newest. Change the sort order by clicking the column headers. Clicking the selected sort column a second time sorts the data in the opposite direction. The small triangle pointing up or down next to a column header indicates the sort direction and marks that column as the sort column.

Update the Package in an AppExchange Listing

If you add features to a published app or component, update your AppExchange listing so that new customers get access to the latest version. You can only associate an approved package version with your public listing. If your offering passed the security review within the last year, the new version is auto-approved. The package version must share the same namespace as the version that passed the review.

Note: If the last security review was completed more than a year ago, the security review team may contact you to arrange a new review. Until then, you can continue to list the newer version.

1. Upload the new version of your package to the AppExchange.
2. Log in to the Partner Community.
3. On the Publishing page, click the **Packages** tab. If you developed the new package in the same organization as the previous version, the new package displays automatically. If you developed it in a different organization, first connect the organization that contains the new package on the **Organizations** tab.
4. Find the new package, and then click **Start Review**.
5. Fill out the self-evaluation questionnaire and click **Submit**. If your offering passed the security review within the last year, the new package is auto-approved, and its status changes to Passed; the status change can take up to 24 hours.
6. After your app or component is approved, navigate to the **Listings** tab and click the listing that you want to edit. This opens the AppExchange publishing console.
7. If you're updating an app, click the **App** tab. If you're updating a component, click the **Component** tab.
8. Click **Select Package**, and then find the new package you want to associate with the listing.

9. Click **Save**.

AppExchange FAQ

The following is a list of frequently asked questions about selling on the AppExchange.

- Can I add more industries?
- Do I need an APO to publish my app or component on the AppExchange?
- Can I change my company name?
- Can I create my app or component on a Salesforce sandbox and upload it to the AppExchange?
- Can I edit a review?
- Can I keep the same listing but change the package it provides?
- Can I update my app or component with a new version or patch?
- How Do Customers Find My Listing?
- How do I edit a package after I've created a listing?
- How do I get an API token for my app?
- How do I increase my listing's popularity?
- How do I offer a free trial of my app or component?
- How do I see listings that Salesforce removed?
- How do I upgrade my customers to a new version?
- What's the difference between a free trial and test drive?
- Where can I share my ideas?
- Where can I write a review?

Can I add more industries?

No. To prevent abuse, you can only specify two industries for each listing. If you cover more industries, mention them in the full or brief description of your listing.

Do I need an APO to publish my app or component on the AppExchange?

No, you no longer need an AppExchange Publishing Organization (APO) to publish your app or component on the AppExchange. You can now connect the organization where you developed the app or component directly to the AppExchange publishing console. To connect an organization, open the Publishing page in the Partner Community, and click the **Organizations** tab. Before connecting an organization, make sure that you have the "Manage Listings" permission in the Partner Community.

Can I change my company name?

Yes, you can change your company name and other aspects of your company profile. Open the Publishing page in the Partner Community, and navigate to the **Company Info** tab. You can change the company name, upload a logo, and modify other details on your company profile.

Can I create my app or component on a Salesforce sandbox and upload it to the AppExchange?

No. You can use a sandbox to install and test your app or component, but you must create and upload it using a Developer edition organization.

Can I edit a review?

You can edit reviews that you authored. You can comment on reviews that you did not write.

Can I keep the same listing but change the package it provides?

Yes, you can change the packages that are linked to your listing. First, make sure that you've uploaded the new package and, if the listing is public, that the package has passed the security review.

On the Publishing page in the Partner Community, navigate to the **Packages** tab and find the package associated with the listing that you want to update. Click **Edit Listing** to open the publishing console. If you're updating an app, you can add a package on the **App** tab. If you're updating a component, add it on the **Component** tab.

Can I update my app or component with a new version or patch?

Yes, but you must submit the new package for an AppExchange Security Review and register the package with your License Management App (LMA).

How Do Customers Find My Listing?

Customers can find your app, component, or consulting service in several ways. On the AppExchange, they can search using keywords or browse using categories. They can also find your listing via an external search provider, such as Google. Knowing how your listing is ranked in each of these scenarios helps you get the most visibility with potential customers.

- Most of the time, people look for apps, components, and consulting services by searching for a term (keyword) on the AppExchange home page. The AppExchange returns matching results and sorts them based on keyword relevance. Here are some tips on how this works.
  - If you include a keyword anywhere in your listing, your listing appears in the search results for that keyword.
  - Generally, a keyword's relevance is increased by appearing earlier in the listing.
  - Generally, a keyword's relevance is increased by appearing more than once in the listing. Note that listing a keyword multiple times in a row does not improve the listing's ranking.
  - When two or more keywords are searched, only listings with all keywords in the same order are returned. In addition, searches for multiple keywords also match camel-cased words (for example, a search for "Great App" matches "GreatApp").
  - Listings at the top of a search are the most relevant from both a keyword and popularity perspective.
- When people look for apps, components, or consulting services by browsing categories, the listings in a category are sorted based on popularity during the past 30 days. Popularity is based on the actions customers can take, such as installing an app or component, taking a test drive, watching a demo or other resources, and clicking the **Learn More** link. Activities that show greater commitment, such as installing an app or component, are weighted more heavily than activities showing less commitment, such as clicking screenshots. The number of reviews and the average rating on a listing do not contribute to the popularity of the listing.

- People can also sort results by rating, release date, name, or provider name. Search results ranked by rating are sorted first by the number of stars and then by the number of reviews. For example, a listing with 1 review and five stars is ranked above a listing with 20 four-star reviews.
- Because the AppExchange is a public website, search engines index your listing pages and return them in their search results. To improve your ranking with external search providers, make sure that you cross reference your listing URL on your website, blog, Facebook, and Twitter pages.

How do I edit a package after I've created a listing?

Log in to the Partner Community and navigate to the AppExchange Publishing page. Click the **Packages** tab to view a list of all packages uploaded to the AppExchange. From this list you can:

- Search for a package by keyword.
- Select **Unlisted Packages** from the drop-down list to see only the packages that haven't yet been linked to a listing.
- Click **Start Review** to begin the security review process.
- For a listed package, click **Edit Listing** to edit listing details, such as pricing information, banners, and logos.
- For an app or component in a managed package, click **Manage Licenses** to update the license settings for this package version, such as whether your offering is free or for sale, if and when it expires, and how many people in the installer's organization can access it.

How do I get an API token for my app?

You can request an API token for your app after it passes the AppExchange security review. To request a token, log a case in the Partner Community under the **AppExchange and Feature Requests** > **API Token Request** category. Specify the type of token (SOAP) and if you're using OAuth.

Note: This feature is available to eligible partners. For more information on the Partner Program, including eligibility requirements, visit www.salesforce.com/partners.

How do I increase my listing's popularity?

Popularity is based on customer activity. The AppExchange measures everything users do on your listing: install, learn more, test drive, demo, view screenshots, white papers, or data sheets, and more. The AppExchange weighs the activity according to its level of importance as indications of interest and filters out attempts to abuse the system.

The AppExchange recalculates popularity daily and then summarizes and evaluates results over 30 days. When you browse by category, you see listings sorted by their relative popularity over the past 30 days.

How customers have reviewed or rated the listing does not affect popularity. AppExchange visitors can sort by rating if they're interested.

Here are a few hints on improving rankings.

- Include a test drive. People like being able to try out an app or component. The number of test drives influences popularity. You also get the added benefit of being able to collect leads.
- Add images. One of the first things that visitors do is click the **View Screenshots** button. Many people don't even look at a listing that doesn't have screenshots.
- Add resources that demonstrate how your app or component affects the customer's bottom line. For example, if you have research showing that a component helps support representatives resolve cases faster, include that information in a data sheet.
- Be up front with your pricing. If you don't include pricing on your listing, people become disinterested quickly.

How do I offer a free trial of my app or component?

When you're creating or editing a listing, the **Trials** tab asks whether you want to offer a free trial or test drive. A free trial lets customers try your app in an interactive organization that you've customized. A test drive lets customers try a read-only version of your app without logging in to Salesforce. For more information, see Provide a Free Trial of Your Solution on page 248.

How do I see listings that Salesforce removed?

The AppExchange doesn't allow you to view listings that Salesforce removed. However, you can view private listings, which can include listings removed by Salesforce, usually because of problems discovered during the periodic security review. To view your private listings, on the Publishing page, navigate to the **Listings** tab. Click **Private Listings** from the drop-down list.

How do I upgrade my customers to a new version?

Create a new version of your managed package and upload it in the released state. After you upload, you can share the Install URL with your existing customers so that they can upgrade. If you're deploying only a bug fix to your customers and want to upgrade them automatically, see "Scheduling Push Upgrades" in the Salesforce online help. You can use the License Management App (LMA) to find out which customers need to upgrade.

Customers can also check whether an upgrade is available by logging in to the AppExchange and viewing the My Account page. If a new version of the app or component is available, it appears on this page.

What's the difference between a free trial and test drive?

When you're creating or editing a listing, the **Trials** tab asks whether you want to offer a free trial of Salesforce and your app or component. The free trial is a non-production Salesforce organization that includes your package and sample data. If a customer chooses to purchase the app or component instead of letting the trial expire, the organization becomes a production version. We recommend that you write a data cleanup script and include a button in your app or component that gives customers the option to remove sample data.

You can also choose to offer a test drive, which is a read-only version of your app or component that all customers taking the test drive log in to. Like a free trial, a test drive uses Developer Edition organizations that include sample data and whatever configuration options you choose.

Where can I share my ideas?

You can share your ideas on how to improve the AppExchange or Salesforce partner programs in the Collaboration section of the Partner Community. These ideas are only seen by Salesforce and other partners. To share ideas more publicly, please post them on the IdeaExchange.

Where can I write a review?

On the listing page, click the number of reviews or **Write the first**. If there are already reviews, you are directed to the review page where you can click **Write a review**. Each user can write only one review per listing.

Important: You cannot write a review for your own listing. Please review the Terms of Use for AppExchange for additional legal information.

Can I have multiple listings for an app or component?

No, you can associate an app or component with only one listing. In addition, you can't duplicate a package (or create a new package version) just to list the app or component in a new listing. This behavior is to your advantage, because it's easier for you to maintain and upgrade the app or component over its lifecycle. It also helps your listing achieve a higher ranking in the AppExchange, because the metrics that Salesforce uses to rank apps and components, like page views, aren't diluted across multiple listings.

CHAPTER 8 Manage Orders

In this chapter ...

- Channel Order App
- Set Up the Channel Order App
- Upgrade the Channel Order App
- Manage Orders in the Channel Order App
- Channel Order Apex API

Create, manage, and submit orders to the Partner Operations team with the Channel Order App (COA). If you're an OEM partner, you can use the COA to provision Salesforce licenses and for revenue sharing. If you're an ISVforce partner, you can use the COA for revenue sharing.

The COA is pre-installed in your Partner Business Org, but before you use it, you must complete the training offered by the Partner Operations team. Acquire your Partner Business Org, pass the solution security review, and then sign up for COA training.

To sign up, log a case in the Partner Community. For the case topic, select **Channel Order Application (COA)**, and then select **Create a Case**.

Note: Submit orders based on the sales and licensing of your solutions to customers, as required by your partner agreement.

Channel Order App

When a customer buys your AppExchange product or requests changes to a subscription, submit an order with the Channel Order App (COA). After Salesforce receives your order, we activate or provision the product in the customer's org and invoice you based on the terms of your partnership agreement.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Note: The COA is available in English to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, visit https://partners.salesforce.com.

With the COA, you can:

- Submit initial orders for new customers
- Submit add-on, upgrade, renewal, reduction, and cancellation orders for existing customers
- Edit, recall, and clone orders that you've submitted
- Delete order drafts
- View details about your customers, such as order history

To comply with your revenue-sharing agreement, submit an order after every customer transaction. The information that you provide keeps our records up-to-date and ensures that the invoices you receive are accurate. For questions about your agreement, log a case in the Partner Community.

Tip: If you're a new AppExchange partner, stop by Trailhead and earn the Channel Order App Basics badge before you get started with the COA. You'll learn how order submission fits into the partnership experience and have an opportunity to test your knowledge.

Channel Order App Objects

Before you start working with the Channel Order App (COA), learn about the app's objects. Understanding what the objects contain makes it easier to create accurate orders that are processed quickly by Salesforce.

Order Types

When you create an order in the Channel Order App (COA), you choose an order type that tells Salesforce how to process the products on the order. Learn how to select the correct type based on your customer's needs.

Order Status

After you create an order in the Channel Order App (COA), Salesforce assigns an order status to help you track progress and, if needed, resolve issues. Order status also determines the actions you can perform on an order, like editing or cloning.

Channel Order App Permission Sets

You control access to the Channel Order App (COA) with the COA User and COA Admin user permission sets. The permission sets determine how users can interact with objects and features in the COA. Learn how to assign the correct permission set based on a user's role on your team.

Channel Order App Objects

Before you start working with the Channel Order App (COA), learn about the app's objects. Understanding what the objects contain makes it easier to create accurate orders that are processed quickly by Salesforce.

Name	Description
Customer	Contains details about a customer who's purchased your product, such as the billing address and Salesforce org ID.

Name	Description
	When you create an initial order in the order submission wizard, the COA creates a customer record using the customer information that you provide.
Partner Product Catalog	Contains a product in your catalog that you can sell to customers. For example, more API calls or an increase in storage in the customer's org. Salesforce configures the products in your catalog based on your partnership agreement. During setup, you import your catalog to the COA. Unless permitted by your agreement, you can't edit your product catalog.
Partner Contract Terms	Contains the contract terms that apply to a product. For example, the default length of a contract and how often a customer is billed. Salesforce configures contract terms based on your partnership agreement. During setup, you import the terms to the COA. Unless permitted by your agreement, you can't edit your contract terms.
Service Order	Contains information about an order that you're submitting to Salesforce. For example, the date the customer signed the Salesforce agreement. When you create an order in the order submission wizard, the COA creates a service order automatically.
Service Order Detail	Contains deal-specific information about a product line item on an order. For example, the number of licenses the customer is buying and the price per license. When you add products to an order in the order submission wizard, the COA configures service order details automatically. You can't access service order detail records directly unless you submit orders with the Channel Order Apex API.

To understand how these objects fit together, let's look at an example.

You sell a human resources app on AppExchange, and a new customer decides to buy some licenses. After you work out the terms of the purchase with the customer, you use the License Management App (LMA) to provision the licenses in their org. Then you submit an order in the COA to tell Salesforce about the sale.

1. On the Service Orders tab, you launch the order submission wizard. The COA creates a service order record.
2. You provide details about the customer, like the billing address. The COA uses these details to create a customer record. In the future, if the customer requests changes to the subscription, you can look up and reuse the details that you provided.
3. You select the contract terms that apply to the order. The COA looks up the corresponding partner contract terms record.
4. You select the product from your catalog that you sold. The COA looks up the corresponding partner product catalog record.
5. You tell us how many licenses you sold and for how much. The COA configures the service order details for the order.
6. You select a start date, review the order, and submit it to Salesforce for invoicing. The COA adds the service order record to the list of existing orders.

Other Channel Order App Objects

Salesforce uses other objects to help process and manage your orders or to assist with debugging. Most of the time, you don't see or interact with these objects.

Name	Description
Customer Order Product History	Contains deal-specific information about an active product on an order, along with the corresponding customer details. After Salesforce activates or provisions an order, we create a customer order product history record for each product on the order. These records become part of the customer's order history, which includes all active products associated with the customer. You can't access customer order product history records directly. To see a customer's order history, open the customer record in the COA and go the Products related list.
Partner Pricebook Entry	Contains one or more products from a catalog. Salesforce uses partner pricebook entries to organize your product catalog. Unless you receive instructions from us, don't modify the partner pricebook entries in your org.
Service Order Log	Stores information about the performance of the COA for debugging purposes. Salesforce uses service order logs to troubleshoot issues with the COA. Unless you receive instructions from us, don't modify the service order logs in your org.

Order Types

When you create an order in the Channel Order App (COA), you choose an order type that tells Salesforce how to process the products on the order. Learn how to select the correct type based on your customer's needs.

Note: Your agreement with Salesforce determines the order types available to you. You might not be able to submit every order type.

Order type reflects the stage of your relationship with the customer: beginning, middle, or end. Order type also determines when we activate or provision the order for the customer.

Type	Stage	Use To	Effective Date
Initial	Beginning	Submit a first order for a new customer.	The service start date you specify on the order.
Add On	Middle	Add products or increase the number of licenses on a customer contract.	The service start date you specify on the order.
Upgrade	Middle	Increase the quantity and price of licenses mid-contract, or upgrade a customer to a higher-priced product mid-contract.	The service start date you specify on the order.
Reduction	Middle	Remove products, or decrease the number of licenses on a customer contract.	The customer's contract renewal date. Notify Salesforce of the reduction according to the terms of your partnership agreement,

Type	Stage	Use To	Effective Date
			usually at least 30 days before a contract renews. You can't submit a reduction order within 5 days of a contract renewal date.
Renewal	Middle	Renew a contract that isn't set to auto-renew, or change the price of existing products on contract renewal.	The customer's contract renewal date.
Cancellation	End	End a contract with a customer and cancel all products. A cancellation order permanently removes your products from the customer's org.	The customer's contract renewal date. Notify Salesforce of the cancellation according to the terms of your partnership agreement, usually at least 30 days before a contract renews. You can't submit a cancellation order within 5 days of a contract renewal date.

Order Status

After you create an order in the Channel Order App (COA), Salesforce assigns an order status to help you track progress and, if needed, resolve issues. Order status also determines the actions you can perform on an order, like editing or cloning.

Here's how we assign order status.

Status	Assigned When
Draft	You save your order, but don't submit it to Salesforce. After you create, clone, or recall an order, its status is Draft by default.
Received	Salesforce receives your order, but hasn't started processing it. You have 2 hours from the time Salesforce receives the order to recall it and edit products, license quantities, and pricing.
In Process	Salesforce is reviewing and processing your order.
Activated	Salesforce has processed your order and is ready to invoice you for revenue sharing. This status applies to: • ISVforce orders that don't provision licenses in a customer's org • Processed OEM orders that have a future start date • All OEM and ISVforce cancellation and reduction orders
Provisioned	Salesforce has processed your order, provisioned licenses in the customer's org, and is ready to invoice you for revenue sharing. This status applies only to OEM orders that provision licenses in the customer's org.

Status	Assigned When
Error	Salesforce encounters an issue that prevents us from processing your order. We return the order and ask you to fix the issue before resubmitting.

Order status determines what you can do with the order. Here are the actions that you can perform for each order status.

	Possible Actions				
Order Status	**Edit**	**Recall**	**Delete**	**Submit**	**Clone**
Draft	*		*	*	*
Received	*	*			*
In Process					*
Activated					*
Provisioned					*
Error					*

Channel Order App Permission Sets

You control access to the Channel Order App (COA) with the COA User and COA Admin user permission sets. The permission sets determine how users can interact with objects and features in the COA. Learn how to assign the correct permission set based on a user's role on your team.

Permission Set	Users Can	Assign To
COA User	Create and manage customers. Submit, edit, recall, and clone orders, and delete order drafts. View COA custom objects.	Team members who submit and manage customer orders.
COA Admin User	Create and manage customers. Submit, edit, recall, and clone orders, and delete order drafts. Configure whether orders are sent to Salesforce or a test environment. Modify COA custom objects.	Team members who administer the COA and whose role includes these tasks: • Setting up the COA • Assigning access to the COA • Building custom integrations using COA objects • Serving as the context user for the COA email service

Set Up the Channel Order App

Install the Channel Order App (COA) and get it ready to sync your product data from Salesforce. After the app is configured, provide access to the right people on your team by assigning permission sets. Then configure a tab to display customer information, such as order history and related products.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

1. Install the Channel Order App
   Install the Channel Order App (COA) in the Salesforce org where you manage licenses for your products, usually your Partner Business Org. If you're an existing partner and the COA is already installed in your org, you don't need to reinstall the app to receive upgrades. Salesforce pushes new versions of the app to your org.
2. Assign Permission Sets to Channel Order App Users
   Assign a Channel Order App (COA) permission set to give team members access to the app. Assign the COA User permission set to users who submit and manage customer orders. Assign the COA Admin User permission to users who need full access to the app's objects and features, including the ability to set up a connection to Salesforce.
3. Define a Channel Order App Email Service
   After you assign Channel Order App (COA) permission sets, define an email service to make your org ready to sync your product catalog.
4. Connect the Channel Order App to Salesforce
   After you install the Channel Order App (COA), connect the app to Salesforce and import your product catalog. Your product catalog includes the products that you can sell and the contract terms that apply to your orders. After the connection is configured, Salesforce pushes catalog updates to your org.
5. Display Customers in the Channel Order App
   After you install the Channel Order App (COA), create a custom tab to display customer information.
6. Assign Page Layouts in the Channel Order App
   After you install the Channel Order App (COA), assign a custom page layout to the customer object.

Install the Channel Order App

Install the Channel Order App (COA) in the Salesforce org where you manage licenses for your products, usually your Partner Business Org. If you're an existing partner and the COA is already installed in your org, you don't need to reinstall the app to receive upgrades. Salesforce pushes new versions of the app to your org.

USER PERMISSIONS

To install packages:
- Download AppExchange Packages

1. Log in to AppExchange using the credentials of the org where you want to install the COA.
2. Go to the AppExchange listing for the COA: https://appexchange.salesforce.com/listingDetail?listingId=a0N300000055ailEAA.
3. Click **Get It Now**.
4. Click **Install in Production**.
5. Agree to the Terms & Conditions, and click **Confirm and Install**.
6. Log in to the org where you want to install the COA.
7. Review the package installation details, and click **Continue**.
8. Approve access by third-party websites, and click **Continue**.

9. Review the API access requirements for the package, and click **Next**.
10. Grant access to package contents, and click **Next**.

    Note: Salesforce recommends granting access only to admins and assigning access to other users as needed after the app is installed.

11. Click **Install**.
12. After the installation completes, go to the App Launcher and confirm that Partner Order appears in the list of available apps.

Assign Permission Sets to Channel Order App Users

Assign a Channel Order App (COA) permission set to give team members access to the app. Assign the COA User permission set to users who submit and manage customer orders. Assign the COA Admin User permission to users who need full access to the app's objects and features, including the ability to set up a connection to Salesforce.

USER PERMISSIONS

To assign a permission set:
- Assign Permissions Sets

1. Log in to the org where the COA is installed.
2. From Setup, enter `Users` in the Quick Find box, then click **Users**.
3. Select a user.
4. In the Permission Set Assignments related list, click **Edit Assignments**.
5. Select the COA User or COA Admin User permission set, and click **Add**.
6. Click **Save**.

Define a Channel Order App Email Service

After you assign Channel Order App (COA) permission sets, define an email service to make your org ready to sync your product catalog.

USER PERMISSIONS

To configure Apex email services and email service addresses:
- Modify All Data

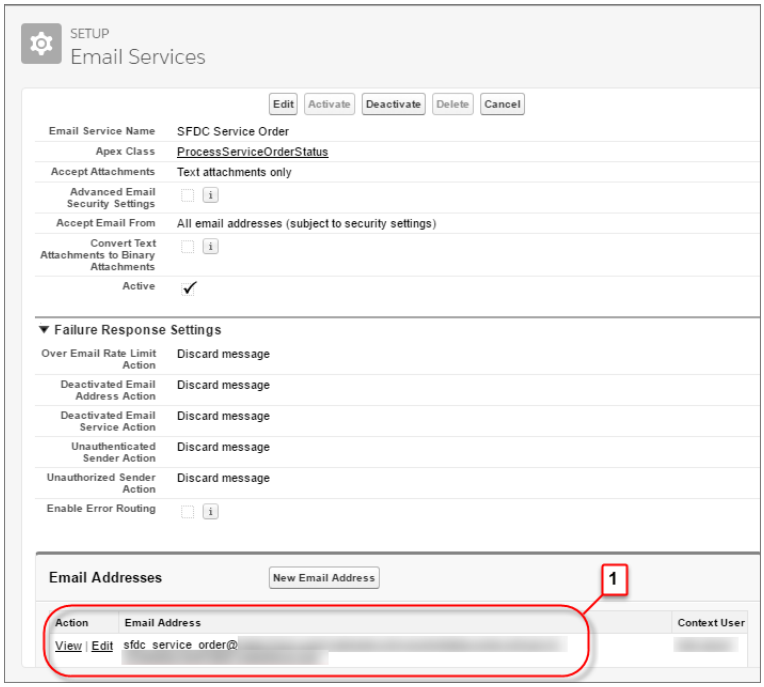
1. Log in to the org where the COA is installed.
2. From Setup, enter `Email Services` in the Quick Find box, then click **Email Services**.
3. Click **New Email Service**.
4. Specify values for the following fields. Leave the other fields as is.

Field	Value
Email Service Name	`SFDC Service Order`
Apex Class	`ProcessServiceOrderStatus`
Accept Attachments	**Text attachments only**
Active	Select to enable

5. Click **Save and New Email Address**.
6. Specify values for the following fields. Leave the other fields as is.

Field	Value
Email Service Name	`SFDC_Service_Order`
Active	Select to enable
Context User	Select a Salesforce admin in your org

7. For **Accept Email From**, remove the autopopulated email address. This field must be blank. Otherwise, the email service can't connect to Salesforce.
8. Click **Save**. Salesforce generates a unique address for the email service (1), which the COA uses to sync your product data.



9. Confirm that the COA Admin User permission set is assigned to the email service's context user. If the context user doesn't have this permission set, assign it to them.

Connect the Channel Order App to Salesforce

After you install the Channel Order App (COA), connect the app to Salesforce and import your product catalog. Your product catalog includes the products that you can sell and the contract terms that apply to your orders. After the connection is configured, Salesforce pushes catalog updates to your org.

Tip: Before you configure your connection, make sure that you have credentials for your COA production connection. These credentials are unique to your company and are provided to you by Salesforce. If you don't have credentials, log a case in the Partner Community.

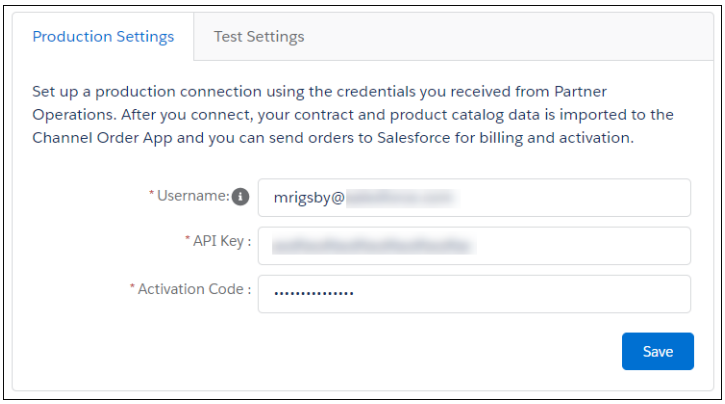
USER PERMISSIONS

To manage custom apps:
- Customize Application

To import product data:
- COA Admin User

1. Log in to the org where the COA is installed.
2. Open the App Launcher.

3. Under All Items, click **COA Setup**.
4. Go to Production Settings, and provide your username, API key, and activation code.



5. Click **Save**.
   The COA imports your product catalog and contract terms.

Display Customers in the Channel Order App

After you install the Channel Order App (COA), create a custom tab to display customer information.

USER PERMISSIONS

To create and edit custom tabs:
- Customize Application

1. Log in to the org where the COA is installed.
2. From Setup, enter `Tabs` in the Quick Find box, then click **Tabs**.
3. In the Custom Object Tabs related list, click **New**.
4. Specify values for the following fields. Leave the other fields as is.

Field	Value
`Object`	**Customer**
`Tab Style`	Select your preferred tab style

5. Click **Next**.
6. Select the user profiles for which the tab is available, and click **Next**.
7. Add the tab to the Partner Order custom app.
8. Click **Save**.

Assign Page Layouts in the Channel Order App

After you install the Channel Order App (COA), assign a custom page layout to the customer object.

USER PERMISSIONS

To create and edit custom objects:
- Customize Application

1. Log in to the org where the COA is installed.
2. From Setup, enter `Object Manager` in the Quick Find box, then click **Object Manager**.
3. Click **Customer**.

4. Click **Page Layouts**.
5. Click **Page Layout Assignment**.
6. Click **Edit Assignment**.
7. Select at least one profile.
8. From the list of available layouts, choose **COA Customer Layout**.
9. Click **Save**.

Upgrade the Channel Order App

If you've installed a previous version of the Channel Order App (COA), Salesforce pushes new versions to your org as they become available. Before you install an upgrade, review the considerations to understand how customizations in your org could be affected. Depending on the COA version you use, some additional configuration might be required after upgrading.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Channel Order App Upgrade Considerations

Before you install a new version of the Channel Order App (COA), understand what's changed in the app and how the changes can affect your customizations.

Upgrade the Channel Order App

Follow these steps to upgrade an earlier version of the Channel Order App (COA) to v2 and later.

Field Mapping in Channel Order App v2 and Later

In Channel Order App (COA) v2, we retired some fields on the service order detail object. If you're upgrading from v1.39 or earlier to v2 or later, the table shows how the retired fields map to new ones.

Channel Order App Upgrade Considerations

Before you install a new version of the Channel Order App (COA), understand what's changed in the app and how the changes can affect your customizations.

Upgrades from v1.39 or Earlier to v2

If you're using COA v1.39 or earlier, the following considerations apply when upgrading to v2 or later.

Replaced Service Order Credentials Page

In v2 and later, the COA Setup page replaces the Service Order Credentials page. After you upgrade, go to the setup page and refresh your connection to Salesforce. If the connection isn't refreshed, Salesforce can't receive your orders.

New Permission Sets for Accessing the COA

In v1.39 and earlier, a custom profile controls access to the COA. In v2 and later, you control access with permission sets. After you upgrade, assign a permission set to the people on your team who use the COA, including those who accessed the app using the custom profile. Without a permission set, your users can't access the COA.

New Customers Tab

In v2 and later, the new Customers tab shows you customer information, including order history and related products. After you upgrade, you must create this tab and configure it to display in the app.

Replaced Orders Tab

In v2 and later, the Service Orders tab replaces the Orders tab. After you upgrade, remove the Orders tab from the app and configure the Service Orders tab.

Updated Page Layouts

In v2 and later, the customer, service order, partner contract terms, and partner product catalog objects have updated page layouts. After you upgrade, assign the updated layouts to each object.

Replaced Partner Order Submit API

In v2 and later, the Channel Order API replaces the Partner Order Submit API. When you upgrade, you can still submit orders using the Partner Order Submit API, and your existing integrations continue to function. However, the Partner Order Submit API doesn't include features introduced in the Channel Order Apex API, such as the ability to edit, recall, and clone orders.

Other Changes to the API

We changed how the API sets the status of submitted orders. In v1.39 and earlier, the Partner Order API automatically updated the `Service_Order_Status__c` field of a submitted order. In v2 and later, the Channel Order API provides a response that reports if the submit operation succeeded, but doesn't update `Service_Order_Status__c` field.

Upgrade the Channel Order App

Follow these steps to upgrade an earlier version of the Channel Order App (COA) to v2 and later.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

1. Assign Permission Sets to Channel Order App Users

   If you're upgrading to Channel Order App (COA) v2 and later, assign permission sets to give team members access to the app. Assign the COA User permission set to users who submit and manage customer orders. Assign the COA Admin User permission to users who need full access to the app's objects and features, including the ability to set up a connection to Salesforce.

2. Display Customers in the Channel Order App

   If you're upgrading to Channel Order App (COA) v2 and later, create a custom tab to display customer information in the app.

3. Display Service Orders in the Channel Order App

   If you're upgrading to Channel Order App (COA) v2 and later, remove the existing Orders tab and replace it with the new Service Orders tab.

4. Update Page Layouts in the Channel Order App

   If you're upgrading to Channel Order App (COA) v2 and later, assign updated page layouts to the customer, service order, partner contract terms, and partner product catalog objects.

5. Refresh the Channel Order App's Connection to Salesforce

   If you're upgrading the Channel Order App (COA) to v2 or later, refresh your production connection to Salesforce. After your connection refreshes, you can submit orders to Salesforce.

Assign Permission Sets to Channel Order App Users

If you're upgrading to Channel Order App (COA) v2 and later, assign permission sets to give team members access to the app. Assign the COA User permission set to users who submit and manage customer orders. Assign the COA Admin User permission to users who need full access to the app's objects and features, including the ability to set up a connection to Salesforce.

USER PERMISSIONS

To assign a permission set:
- Assign Permissions Sets

1. Log in to the org where the COA is installed.

2. From Setup, enter `Users` in the Quick Find box, then click **Users**.
3. Select a user.
4. In the Permission Set Assignments related list, click **Edit Assignments**.
5. Select the COA User or COA Admin User permission set, and click **Add**.
6. Click **Save**.

Display Customers in the Channel Order App

If you're upgrading to Channel Order App (COA) v2 and later, create a custom tab to display customer information in the app.

USER PERMISSIONS

To create and edit custom tabs:
- Customize Application

1. Log in to the org where the COA is installed.
2. From Setup, enter `Tabs` in the Quick Find box, then click **Tabs**.
3. In the Custom Object Tabs related list, click **New**.
4. Specify values for the following fields. Leave the other fields as is.

Field	Value
`Object`	**Customer**
`Tab Style`	Select your preferred tab style

5. Click **Next**.
6. Select the user profiles for which the tab is available, and click **Next**.
7. Add the tab to the Partner Order custom app.
8. Click **Save**.

Display Service Orders in the Channel Order App

If you're upgrading to Channel Order App (COA) v2 and later, remove the existing Orders tab and replace it with the new Service Orders tab.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

To manage custom apps:
- Customize Application

1. Log in to the org where the COA is installed.
2. From Setup, enter `App Manager` in the Quick Find box, then click **App Manager**.
3. For Partner Order, click ( ) and select **Edit**.
4. From the Selected Tabs list, remove **Orders**.
5. Add **Service Orders** to the Selected Tabs list.
6. Click **Save**.

Update Page Layouts in the Channel Order App

If you're upgrading to Channel Order App (COA) v2 and later, assign updated page layouts to the customer, service order, partner contract terms, and partner product catalog objects.

1. Log in to the org where the COA is installed.
2. From Setup, enter `Object Manager` in the Quick Find box, then click **Object Manager**.
3. Assign the updated page layout to the customer object.
   a. Click **Customer**.
   b. Click **Page Layouts**.
   c. Click **Page Layout Assignment**.
   d. Click **Edit Assignment**.
   e. Select at least one profile.
   f. From the list of available layouts, choose **COA Customer Layout**.
   g. Click **Save**.
4. Repeat these steps for service order, partner contract terms, and partner product catalog. These objects use the following page layout names.

Object	Page Layout Name
Service Order	Service Order Layout
Partner Contract Terms	Partner Contract Terms Layout
Partner Product Catalog	Partner Product Catalog Layout

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

To create and edit custom objects:
- Customize Application

Refresh the Channel Order App's Connection to Salesforce

If you're upgrading the Channel Order App (COA) to v2 or later, refresh your production connection to Salesforce. After your connection refreshes, you can submit orders to Salesforce.

1. Log in to the org where the COA is installed.
2. Open the App Launcher.
3. Under All Items, click **COA Setup**.
4. Go to Production Settings, and click **Refresh Connection**.
   After you refresh the connection, your order history is imported to the app.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

To manage custom apps:
- Customize Application

To import product data:
- COA Admin User

Field Mapping in Channel Order App v2 and Later

In Channel Order App (COA) v2, we retired some fields on the service order detail object. If you're upgrading from v1.39 or earlier to v2 or later, the table shows how the retired fields map to new ones.

Note: Field names are prefixed with `CHANNEL_ORDERS__` unless otherwise noted.

Fields

Old Field (Retired)	New Field	Notes
Application__c	None	Field retired. This field doesn't populate with data for orders created in COA v2 and later.
Customer_Price__c	Customer_Price_Per_Month__c	Represents the product price per unit per month.
Fixed_Price__c	pc_Fixed_Price__c	Represents the fixed price of the product at the time the order was created.
Floor_Price__c	None	Field retired. This field doesn't populate with data for orders created in COA v2 and later.
Estimated_SFDC_Price_Per_Month__c	SFDC_Price__c	Represents the total amount due to Salesforce based on the estimated value of the product.
Number_Of_Users_ISVforce__c	None	Field retired. This field doesn't populate with data for orders created in COA v2 and later.
pc_Floor_Price__c	None	Field retired. This field doesn't populate with data for orders created in COA v2 and later.
pc_PNR__c	PNR__c	Represents the percent net revenue of the product at the time the order was created.
pc_Pricing_Unit__c	None	Field retired. This field doesn't populate with data for orders created in COA v2 and later.
pc_Product_ID__c	Product_ID__c	Represents the ID of the product.
Pricing_Type__c	pc_Pricing_Type__c	Represents the pricing model of the product.
Product_Line_Desc_Overridden__c	None	Field retired. This field doesn't populate with data for orders created in COA v2 and later.
Special_Instructions__c	None	Field retired. This field doesn't populate with data for orders created in COA v2 and later.

Manage Orders in the Channel Order App

When a customer purchases your AppExchange product or requests changes to a subscription, submit an order to Salesforce. After you create the order, you can edit, recall, or clone it. If the order is a draft, you can delete it.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Submit an Order

Submit an order to Salesforce when a customer purchases new products or requests changes to a subscription. If you're ordering products for a new customer, verify that you have the customer's Salesforce org ID before you create the order.

Edit an Order

You can edit the product, quantity, and pricing details of an order within 2 hours of submitting it to Salesforce. After 2 hours, the order is processed and can't be edited. To change customer details or order type, you must recall the order and create a new one.

Clone an Order

When creating an order that's similar to one you've submitted previously, you can save time by cloning the original order.

Recall an Order

If you don't want Salesforce to process an order you've submitted, recall it. After you recall an order, it becomes read-only, and you can't edit or resubmit it. You can recall an order within 2 hours of submitting it to Salesforce.

Delete a Draft Order

You can delete draft orders that you don't want to submit, like duplicate orders. After you delete a draft order, you can't recover it.

Fix Errors on Returned Orders

If you submit an order that Salesforce can't process, we return the order and ask you to fix the errors that we identified. You can resolve the errors by reading the comments we provide, cloning the returned order, and then submitting the new order with the changes applied.

Submit an Order

Submit an order to Salesforce when a customer purchases new products or requests changes to a subscription. If you're ordering products for a new customer, verify that you have the customer's Salesforce org ID before you create the order.

USER PERMISSIONS

To submit orders:

- COA User

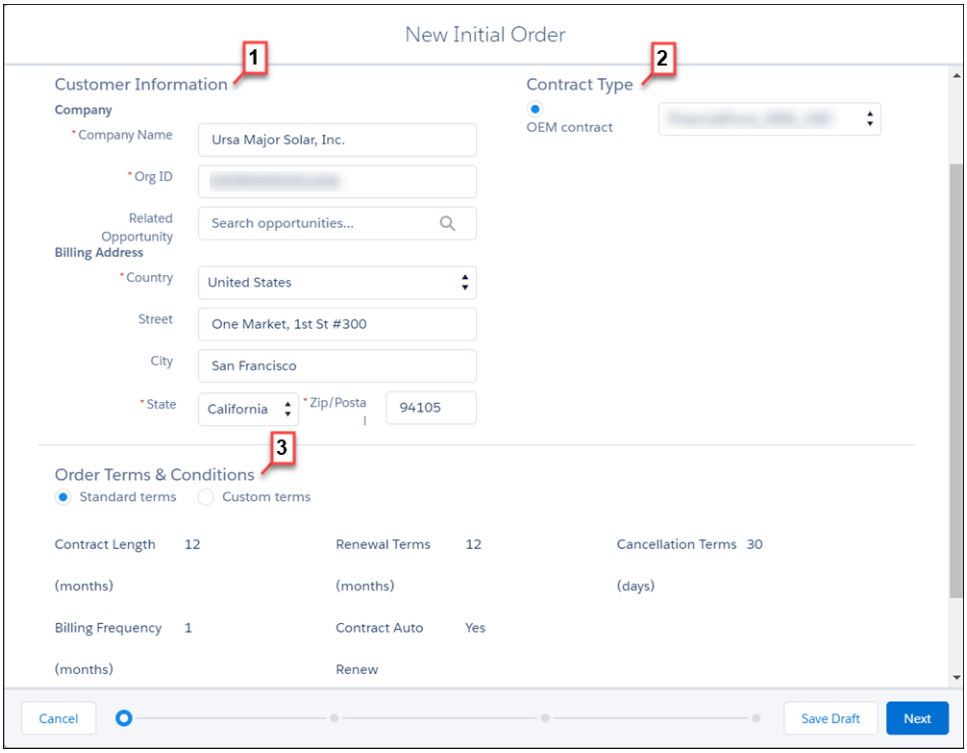
  OR

  COA Admin User

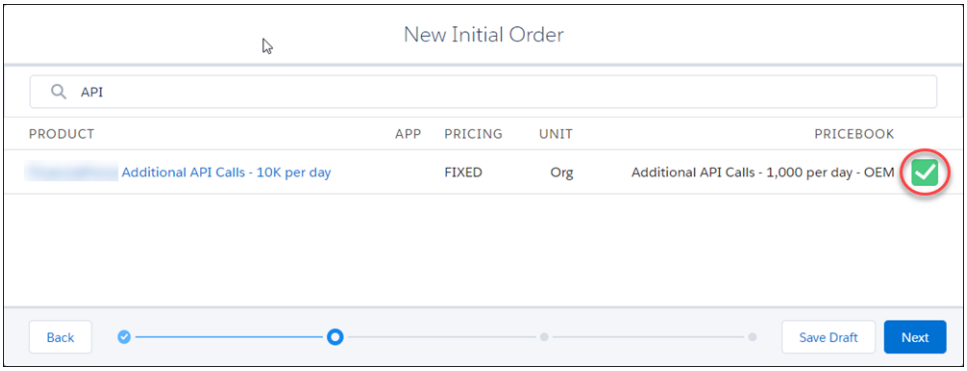
1. Log in to the org where the COA is installed.
2. Open the App Launcher, and click **Partner Order**.
3. On the Service Orders tab, click **New** to open the order submission wizard.
4. Choose **New customer** to create an initial order. Otherwise, choose **Existing customer** and select an order type.

   Tip: If a customer is buying your product for the first time, create an initial order.

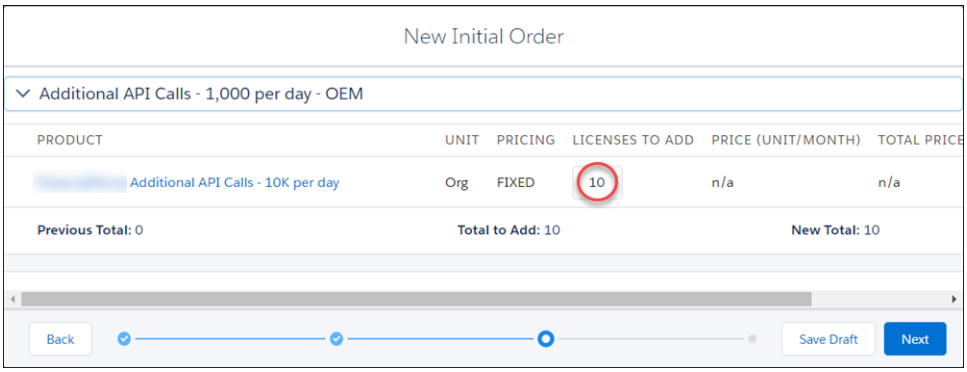
5. Specify customer details (1), contract type (2), and the terms and conditions (3), and then click **Next**.



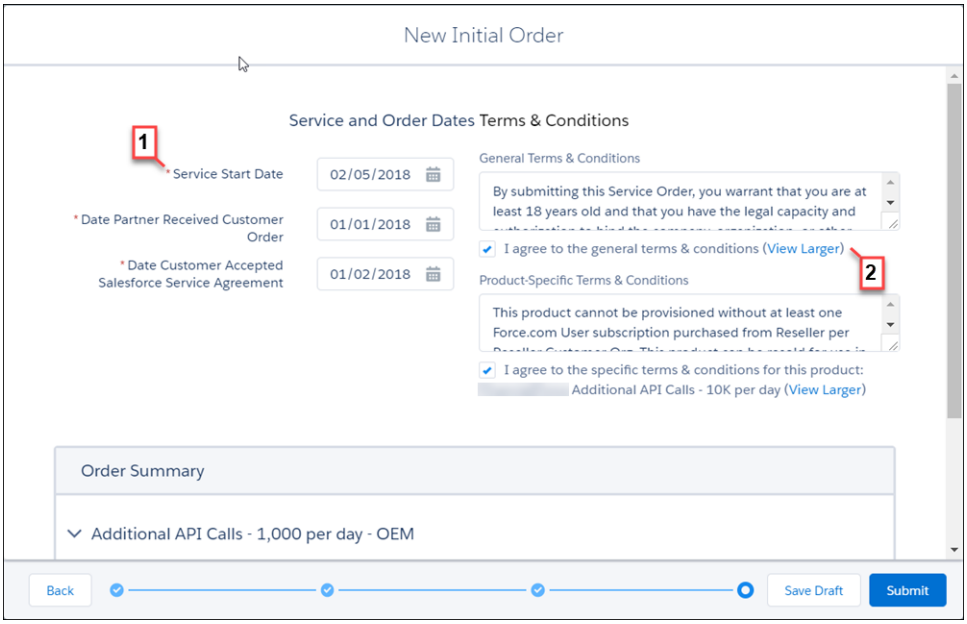
6. Select products for the order, and click **Next**.



7. Adjust the license quantities and, optionally, pricing, and then click **Next**.



8. Enter the service and order dates (1), and then review and accept the terms and conditions (2).



9. Click **Submit**, or save the order as a draft and submit it later.

After you submit an order, it's sent to Salesforce for processing and activation or provisioning. To check the status of an order, go the Service Orders tab.

Edit an Order

You can edit the product, quantity, and pricing details of an order within 2 hours of submitting it to Salesforce. After 2 hours, the order is processed and can't be edited. To change customer details or order type, you must recall the order and create a new one.

1. Log in to the org where the COA is installed.
2. Open the App Launcher, and click **Partner Order**.
3. On the Service Orders tab, find the order you want to edit.

   If you can't find the order, verify that you selected the correct list view.

USER PERMISSIONS

To edit orders:

- COA User

  OR

  COA Admin User

4. Click ( ) and select **Edit**.
5. Update the order's products, quantities, and pricing details, and then click **Resubmit**.

Clone an Order

When creating an order that's similar to one you've submitted previously, you can save time by cloning the original order.

1. Log in to the org where the COA is installed.
2. Open the App Launcher, and click **Partner Order**.
3. On the Service Orders tab, find the order you want to clone.
   If you can't find the order, verify that you selected the correct list view.
4. In the Custom Actions column, click **Clone**.
5. Confirm that you want to clone the order, and click **Continue**.
6. Edit the order as needed, and then click **Save Draft**.

USER PERMISSIONS

To clone orders:
- COA User
  OR
  COA Admin User

Recall an Order

If you don't want Salesforce to process an order you've submitted, recall it. After you recall an order, it becomes read-only, and you can't edit or resubmit it. You can recall an order within 2 hours of submitting it to Salesforce.

1. Log in to the org where the COA is installed.
2. Open the App Launcher, and click **Partner Order**.
3. On the Service Orders tab, find the order that you want to recall.
   If you can't find the order, verify that you selected the correct list view.
4. In the Custom Actions column, click **Recall**.
5. Confirm that you want to recall the order, and click **Continue**.

USER PERMISSIONS

To recall orders:
- COA User
  OR
  COA Admin User

Delete a Draft Order

You can delete draft orders that you don't want to submit, like duplicate orders. After you delete a draft order, you can't recover it.

1. Log in to the org where the COA is installed.
2. Open the App Launcher, and click **Partner Order**.
3. On the Service Orders tab, find the order that you want to delete.
   If you can't find the order, verify that you selected the correct list view.
4. Click ( ) and select **Delete**.
5. Click **Delete** again to confirm.

USER PERMISSIONS

To delete orders:
- COA User
  OR
  COA Admin User

Fix Errors on Returned Orders

If you submit an order that Salesforce can't process, we return the order and ask you to fix the errors that we identified. You can resolve the errors by reading the comments we provide, cloning the returned order, and then submitting the new order with the changes applied.

USER PERMISSIONS

To clone orders:

- COA User

  OR

  COA Admin User

1. Log in to the org where the COA is installed.
2. Open the App Launcher, and click **Partner Order**.
3. On the Service Orders tab, find the returned order.

   If you can't find the order, verify that you selected the correct list view.
4. Click the order, and go to Error Comment to see details about the error.
5. Click **Clone Order**.
6. Apply the requested changes, and then click **Submit**.

If you have trouble resolving the errors, log a case in the Partner Community.

Channel Order Apex API

You can submit orders to Salesforce programmatically using the Channel Order Apex API. To submit an order, use one of the classes provided in the `CHANNEL_ORDERS` namespace.

CHANNEL_ORDERS Namespace

The `CHANNEL_ORDERS` namespace provides classes for submitting orders to Salesforce Partner Operations. After you send an order, you can use other classes in the namespace to edit, recall, or clone the order.

Service Order

Represents an order that you're submitting to Salesforce Partner Operations for processing and activation.

Service Order Detail

Represents an instance of a product on a service order.

Partner Order Submit API

(No longer supported and available only in version 1.39 and earlier of the Channel Order App.) Send orders to Salesforce immediately or asynchronously using the Partner Order Submit API.

CHANNEL_ORDERS Namespace

The `CHANNEL_ORDERS` namespace provides classes for submitting orders to Salesforce Partner Operations. After you send an order, you can use other classes in the namespace to edit, recall, or clone the order.

To use `CHANNEL_ORDERS` namespace classes, you must have Channel Order App v2 or later installed in your Salesforce org. For information on how to invoke methods defined in managed packages, refer to the *Apex Developer Guide*.

The following classes are in the `CHANNEL_ORDERS` namespace.

COA_ServiceOrderSubmit Class

Submit orders to Salesforce Partner Operations for processing and activation.

COA_ServiceOrderEdit Class

Edit orders that you've submitted to Salesforce Partner Operations.

COA_ServiceOrderRecall Class

Recall orders that you've submitted to Salesforce Partner Operations.

COA_ServiceOrderClone Class

Clone an existing order in the org where the Channel Order App (COA) is installed.

COA_ServiceOrderSubmit Class

Submit orders to Salesforce Partner Operations for processing and activation.

Namespace

CHANNEL_ORDERS

Usage

The COA_ServiceOrderSubmit class contains a single `@InvocableMethod` for submitting orders to Salesforce Partner Operations. When invoking a method defined in this class, include the `CHANNEL_ORDERS` namespace prefix:

```
CHANNEL_ORDERS.class.method(args)
```

For details about namespace prefixes or the `@InvocableMethod` annotation, see the *Apex Developer Guide*.

Example

This example receives a list of service orders, submits them, and returns a list of outputs from the submit operation.

Note: For brevity, the methods invoked in this example omit the `CHANNEL_ORDERS` namespace prefix. If you use this code in your implementation, you must include the namespace prefix.

```
public static void submitOrders(List<Service_Order__c> serviceOrders){
   List<COA_ServiceOrderSubmit.COA_ServiceOrderSubmitInput> serviceOrderSubmitInput = new
 List<COA_ServiceOrderSubmit.COA_ServiceOrderSubmitInput>();

    for(Service_Order__c serviceOrder: serviceOrders){
        COA_ServiceOrderSubmit.COA_ServiceOrderSubmitInput input = new
COA_ServiceOrderSubmit.COA_ServiceOrderSubmitInput();
        input.serviceOrderId = serviceOrder.Id;
        serviceOrderSubmitInput.add(input);
    }

    List<COA_ServiceOrderSubmit.COA_ServiceOrderSubmitOutput> serviceOrderSubmitOutputs =
 COA_ServiceOrderSubmit.submit(serviceOrderSubmitInput);

    for(COA_ServiceOrderSubmit.COA_ServiceOrderSubmitOutput serviceOrderSubmitOutput:
serviceOrderSubmitOutputs){
        System.debug('Service Order Id: '+serviceOrderSubmitOutput.serviceOrderId);
        System.debug('Success?: '+serviceOrderSubmitOutput.isSuccess);
        System.debug('Response Messages: '+serviceOrderSubmitOutput.responseMessages);
    }
}
```

Order Status

When you submit a draft order using the COA_ServiceOrderSubmit class, the response tells you if the operation succeeded. The response doesn't set the status of the related service order record, so the `Service_Order_Status__c` field remains `Draft`. If you build an implementation to set the status of submitted orders, we suggest the following logic: if the response includes a success code, set the order status to `Received`. Otherwise, set the status to `Error`. For orders with errors, you can store notes from Salesforce Partner Operations in the `Error_Comment__c` field.

COA_ServiceOrderSubmit Methods

COA_ServiceOrderSubmitInput Class

Wrapper class for input parameters passed to the submit operation.

COA_ServiceOrderSubmitOutput Class

Wrapper class for output parameters returned from the submit operation.

COA_ServiceOrderSubmit Methods

The following are methods for `COA_ServiceOrderSubmit`.

submit(serviceOrderSubmitInput)

Provides an entry point for submitting orders to Salesforce Partner Operations.

`submit(serviceOrderSubmitInput)`

Provides an entry point for submitting orders to Salesforce Partner Operations.

Signature

```
global static List<COA_ServiceOrderSubmit.COA_ServiceOrderSubmitOutput>
submit(List<COA_ServiceOrderSubmit.COA_ServiceOrderSubmitInput> serviceOrderSubmitInput)
```

Parameters

`serviceOrderSubmitInput`

Type: List<COA_ServiceOrderSubmit.COA_ServiceOrderSubmitInput>

List of wrapper classes to pass as input for the submit operation

Return Value

Type: List<COA_ServiceOrderSubmit.COA_ServiceOrderSubmitOutput>

COA_ServiceOrderSubmitInput Class

Wrapper class for input parameters passed to the submit operation.

Namespace

CHANNEL_ORDERS

COA_ServiceOrderSubmitInput Properties

COA_ServiceOrderSubmitInput Properties

The following are properties for `COA_ServiceOrderSubmitInput`.

serviceOrderId
Specifies the ID of the order you are submitting. This field is required.

`serviceOrderId`

Specifies the ID of the order you are submitting. This field is required.

Signature

```
global Id serviceOrderId;
```

Property Value

Type: Id

COA_ServiceOrderSubmitOutput Class

Wrapper class for output parameters returned from the submit operation.

Namespace

CHANNEL_ORDERS

COA_ServiceOrderSubmitOutput Properties

COA_ServiceOrderSubmitOutput Properties

The following are properties for `COA__ServiceOrderSubmitOutput`.

isSuccess
Indicates the success of the submit operation. If true, the order was successfully submitted.

responseMessages
Holds response messages generated by the submit operation.

serviceOrderId
References the order ID passed in by the submit operation.

`isSuccess`

Indicates the success of the submit operation. If true, the order was successfully submitted.

Signature

```
global Boolean isSuccess;
```

Property Value

Type: Boolean

`responseMessages`

Holds response messages generated by the submit operation.

Signature

```
global List<String> responseMessages;
```

Property Value

Type: List<String>

`serviceOrderId`

References the order ID passed in by the submit operation.

Signature

```
global Id serviceOrderId;
```

Property Value

Type: Id

COA_ServiceOrderEdit Class

Edit orders that you've submitted to Salesforce Partner Operations.

Namespace

CHANNEL_ORDERS

Usage

The COA_ServiceOrderEdit class contains a single `@InvocableMethod` for editing orders that have been submitted to Salesforce Partner Operations but haven't been processed. When invoking a method defined in this class, include the `CHANNEL_ORDERS` namespace prefix:

```
CHANNEL_ORDERS.class.method(args)
```

For details about namespace prefixes or the `@InvocableMethod` annotation, see the *Apex Developer Guide*.

Example

This example receives a list of service orders that have been edited, submits them, and returns a list of outputs from the edit operation.

Note: For brevity, the methods invoked in this example omit the `CHANNEL_ORDERS` namespace prefix. If you use this code in your implementation, you must include the namespace prefix.

```
public static void editOrders(List<Service_Order__c> serviceOrders){
    List<COA_ServiceOrderEdit.COA_ServiceOrderEditInput> serviceOrderEditInput = new
List<COA_ServiceOrderEdit.COA_ServiceOrderEditInput>();

    for(Service_Order__c serviceOrder: serviceOrders){
        COA_ServiceOrderEdit.COA_ServiceOrderEditInput input = new
COA_ServiceOrderEdit.COA_ServiceOrderEditInput();
        input.serviceOrderId = serviceOrder.Id;
        serviceOrderEditInput.add(input);
    }

    List<COA_ServiceOrderEdit.COA_ServiceOrderEditOutput> serviceOrderEditOutputs =
COA_ServiceOrderEdit.edit(serviceOrderEditInput);

    for(COA_ServiceOrderEdit.COA_ServiceOrderEditOutput serviceOrderEditOutput:
serviceOrderEditOutputs){
        System.debug('Service Order Id: '+serviceOrderEditOutput.serviceOrderId);
        System.debug('Success?: '+serviceOrderEditOutput.isSuccess);
        System.debug('Response Messages: '+serviceOrderEditOutput.responseMessages);
    }
}
```

COA_ServiceOrderEdit Methods

COA_ServiceOrderEditInput Class

Wrapper class for input parameters passed to the edit operation.

COA_ServiceOrderEditOutput Class

Wrapper class for output parameters returned from the edit operation.

COA_ServiceOrderEdit Methods

The following are methods for `COA_ServiceOrderEdit`.

edit(serviceOrderEditInput)

Provides an entry point to edit orders that you've submitted to Salesforce Partner Operations. You can edit only orders that haven't been processed.

`edit(serviceOrderEditInput)`

Provides an entry point to edit orders that you've submitted to Salesforce Partner Operations. You can edit only orders that haven't been processed.

Signature

```
global static List<COA_ServiceOrderEdit.COA_ServiceOrderEditOutput>
edit(List<COA_ServiceOrderEdit.COA_ServiceOrderEditInput> serviceOrderEditInput)
```

Parameters

`serviceOrderEditInput`

Type: List<COA_ServiceOrderEdit.COA_ServiceOrderEditInput>

List of wrapper classes to pass as input for the edit operation

Return Value

Type: List<COA_ServiceOrderEdit.COA_ServiceOrderEditOutput>

COA_ServiceOrderEditInput Class

Wrapper class for input parameters passed to the edit operation.

Namespace

CHANNEL_ORDERS

COA_ServiceOrderEditInput Properties

COA_ServiceOrderEditInput Properties

The following are properties for `COA_ServiceOrderEditInput`.

serviceOrderId

Specifies the ID of the order you are editing. This field is required.

`serviceOrderId`

Specifies the ID of the order you are editing. This field is required.

Signature

```
global Id serviceOrderId;
```

Property Value

Type: Id

COA_ServiceOrderEditOutput Class

Wrapper class for output parameters returned from the edit operation.

Namespace

CHANNEL_ORDERS

COA_ServiceOrderEditOutput Properties

COA_ServiceOrderEditOutput Properties

The following are properties for `COA_ServiceOrderEditOutput`.

isSuccess
Indicates the success of the edit operation. If true, the order was successfully edited.

responseMessages
Holds response messages generated by the edit operation.

serviceOrderId
References the order ID passed in by the edit operation.

`isSuccess`

Indicates the success of the edit operation. If true, the order was successfully edited.

Signature

```
global Boolean isSuccess;
```

Property Value

Type: Boolean

`responseMessages`

Holds response messages generated by the edit operation.

Signature

```
global List<String> responseMessages;
```

Property Value

Type: List<String>

`serviceOrderId`

References the order ID passed in by the edit operation.

Signature

```
global Id serviceOrderId;
```

Property Value

Type: Id

COA_ServiceOrderRecall Class

Recall orders that you've submitted to Salesforce Partner Operations.

Namespace

CHANNEL_ORDERS

Usage

The COA_ServiceOrderRecall class contains a single `@InvocableMethod` for recalling orders that have been submitted to Salesforce Partner Operations but haven't yet been processed. When you recall an order, it's removed from the processing queue and isn't activated. When invoking a method defined in this class, include the `CHANNEL_ORDERS` namespace prefix:

```
CHANNEL_ORDERS.class.method(args)
```

For details about namespace prefixes or the `@InvocableMethod` annotation, see the *Apex Developer Guide*.

Example

This example receives a list of service orders, recalls them, and returns a list of outputs from the recall operation.

Note: For brevity, the methods invoked in this example omit the `CHANNEL_ORDERS` namespace prefix. If you use this code in your implementation, you must include the namespace prefix.

```
public static void recallOrders(List<Service_Order__c> serviceOrders){
        List<COA_ServiceOrderRecall.COA_ServiceOrderRecallInput> serviceOrderRecallInput
= new List<COA_ServiceOrderRecall.COA_ServiceOrderRecallInput>();

        for(Service_Order__c serviceOrder: serviceOrders){
            COA_ServiceOrderRecall.COA_ServiceOrderRecallInput input = new
COA_ServiceOrderRecall.COA_ServiceOrderRecallInput();
            input.serviceOrderId = serviceOrder.Id;
            serviceOrderRecallInput.add(input);
        }

       List<COA_ServiceOrderRecall.COA_ServiceOrderRecallOutput> serviceOrderRecallOutputs
 = COA_ServiceOrderRecall.recall(serviceOrderRecallInput);

    for(COA_ServiceOrderRecall.COA_ServiceOrderRecallOutput serviceOrderRecallOutput:
serviceOrderRecallOutputs){
        System.debug('Service Order Id: '+serviceOrderRecallOutput.serviceOrderId);
        System.debug('Success?: '+serviceOrderRecallOutput.isSuccess);
        System.debug('Response Messages: '+serviceOrderRecallOutput.responseMessages);
    }
}
```

COA_ServiceOrderRecall Methods

COA_ServiceOrderRecallInput Class

Wrapper class for input parameters passed to the recall operation.

COA_ServiceOrderRecallOutput Class

Wrapper class for output parameters returned from the recall operation.

COA_ServiceOrderRecall Methods

The following are methods for `COA_ServiceOrderRecall`.

recall(serviceOrderRecallInput)

Provides an entry point to recall orders that you've submitted to Salesforce Partner Operations. You can recall only orders that haven't been processed.

`recall(serviceOrderRecallInput)`

Provides an entry point to recall orders that you've submitted to Salesforce Partner Operations. You can recall only orders that haven't been processed.

Signature

```
global static List<COA_ServiceOrderRecall.COA_ServiceOrderRecallOutput>
recall(List<COA_ServiceOrderRecall.COA_ServiceOrderRecallInput> serviceOrderRecallInput)
```

Parameters

`serviceOrderRecallInput`

Type: List<COA_ServiceOrderRecall.COA_ServiceOrderRecallInput>

List of wrapper classes to pass as input for the recall operation

Return Value

Type: List<COA__ServiceOrderRecall.COA__ServiceOrderRecallOutput>

COA_ServiceOrderRecallInput Class

Wrapper class for input parameters passed to the recall operation.

Namespace

CHANNEL_ORDERS

COA_ServiceOrderRecallInput Properties

COA_ServiceOrderRecallInput Properties

The following are properties for `COA_ServiceOrderRecallInput`.

serviceOrderId
Specifies the ID of the order you are recalling. This field is required.

`serviceOrderId`

Specifies the ID of the order you are recalling. This field is required.

Signature

```
global Id serviceOrderId;
```

Property Value

Type: Id

COA_ServiceOrderRecallOutput Class

Wrapper class for output parameters returned from the recall operation.

Namespace

CHANNEL_ORDERS

COA_ServiceOrderRecallOutput Properties

COA_ServiceOrderRecallOutput Properties

The following are properties for `COA_ServiceOrderRecallOutput`.

isSuccess
Indicates the success of the recall operation. If true, the order was successfully recalled.

responseMessages
Holds response messages generated by the recall operation.

serviceOrderId
References the order ID passed in by the recall operation.

`isSuccess`

Indicates the success of the recall operation. If true, the order was successfully recalled.

Signature

```
global Boolean isSuccess;
```

Property Value

Type: Boolean

`responseMessages`

Holds response messages generated by the recall operation.

Signature

```
global List<String> responseMessages;
```

Property Value

Type: List<String>

`serviceOrderId`

References the order ID passed in by the recall operation.

Signature

```
global Id serviceOrderId;
```

Property Value

Type: Id

COA_ServiceOrderClone Class

Clone an existing order in the org where the Channel Order App (COA) is installed.

Note: Only fields that you have permission to create are cloned. DML errors can occur if you don't have sufficient privileges.

Namespace

CHANNEL_ORDERS

Usage

The COA_ServiceOrderClone class contains a single `@InvocableMethod` to clone orders and, optionally, associated line items. When invoking a method defined in this class, include the `CHANNEL_ORDERS` namespace prefix:

```
CHANNEL_ORDERS.class.method(args)
```

For details about namespace prefixes or the `@InvocableMethod` annotation, see the *Apex Developer Guide*.

Example

This example receives a list of service orders, clones them, and returns a list of outputs from the clone operation.

Note: For brevity, the methods invoked in this example omit the `CHANNEL_ORDERS` namespace prefix. If you use this code in your implementation, you must include the namespace prefix.

```
public static void cloneOrders(List<Service_Order__c> serviceOrders){
        List<COA_ServiceOrderClone.COA_ServiceOrderCloneInput> serviceOrderCloneInput =
```

```
new List<COA_ServiceOrderClone.COA_ServiceOrderCloneInput>();

        for(Service_Order__c serviceOrder: serviceOrders){
            COA_ServiceOrderClone.COA_ServiceOrderCloneInput input = new
COA_ServiceOrderClone.COA_ServiceOrderCloneInput();
            input.serviceOrderId = serviceOrder.Id;
            input.cloneProducts = true;
            serviceOrderCloneInput.add(input);
        }

        List<COA_ServiceOrderClone.COA_ServiceOrderCloneOutput> serviceOrderCloneOutputs
= COA_ServiceOrderClone.clone(serviceOrderCloneInput);
        //Further processing of serviceOrderCloneOutputs
    }
```

COA_ServiceOrderClone Methods

COA_ServiceOrderCloneInput Class

Wrapper class for input parameters passed to the clone operation.

COA_ServiceOrderCloneOutput Class

Wrapper class for output parameters returned from the clone operation.

COA_ServiceOrderClone Methods

The following are methods for `COA_ServiceOrderClone`.

clone(serviceOrderCloneInput)

Provides an entry point to clone orders in your org and, optionally, associated line items.

`clone(serviceOrderCloneInput)`

Provides an entry point to clone orders in your org and, optionally, associated line items.

Signature

```
global static List<COA_ServiceOrderClone.COA_ServiceOrderCloneOutput>
edit(List<COA_ServiceOrderClone.COA_ServiceOrderCloneInput> serviceOrderCloneInput)
```

Parameters

`serviceOrderCloneInput`

Type: List<COA_ServiceOrderClone.COA_ServiceOrderCloneInput>

List of wrapper classes to pass as input for the clone operation

Return Value

Type: List<COA__ServiceOrderClone.COA__ServiceOrderCloneOutput>

COA_ServiceOrderCloneInput Class

Wrapper class for input parameters passed to the clone operation.

Namespace

CHANNEL_ORDERS

COA_ServiceOrderCloneInput Properties

COA_ServiceOrderCloneInput Properties

The following are properties for `COA_ServiceOrderCloneInput`.

serviceOrderId
Specifies the ID of the order you are cloning. This field is required.

cloneProducts
Indicates whether to clone the original order's line items. If true, the line items are cloned. This field is required.

`serviceOrderId`

Specifies the ID of the order you are cloning. This field is required.

Signature

```
global Id serviceOrderId;
```

Property Value

Type: Id

`cloneProducts`

Indicates whether to clone the original order's line items. If true, the line items are cloned. This field is required.

Signature

```
global Boolean cloneProducts;
```

Property Value

Type: Boolean

COA_ServiceOrderCloneOutput Class

Wrapper class for output parameters returned from the clone operation.

Namespace

CHANNEL_ORDERS

COA_ServiceOrderCloneOutput Properties

COA_ServiceOrderCloneOutput Properties

The following are properties for `COA__ServiceOrderClone.COA__ServiceOrderCloneOutput`.

isSuccess
Indicates the success of the clone operation. If true, the order was successfully recalled.

responseMessages
Holds response messages generated by the clone operation.

originalServiceOrderId
Specifies the ID of the original order that you cloned.

cloneServiceOrderId
Specifies the ID of the newly created clone order.

`isSuccess`

Indicates the success of the clone operation. If true, the order was successfully recalled.

Signature

```
global Boolean isSuccess;
```

Property Value

Type: Boolean

`responseMessages`

Holds response messages generated by the clone operation.

Signature

```
global List<String> responseMessages;
```

Property Value

Type: List<String>

`originalServiceOrderId`

Specifies the ID of the original order that you cloned.

Signature

```
global Id originalServiceOrderId;
```

Property Value

Type: Id

`cloneServiceOrderId`

Specifies the ID of the newly created clone order.

Signature

```
global Id cloneServiceOrderId;
```

Property Value

Type: Id

Service Order

Represents an order that you're submitting to Salesforce Partner Operations for processing and activation.

Note: Field names are prefixed with `CHANNEL_ORDERS__` unless otherwise noted.

When you submit an order with the Channel Order App API, include the following fields.

Fields

Field	Details
Label Created with New COA **Name** `Created_with_new_COA__c`	**Type** boolean **Properties** Create, Defaulted on create, Filter, Group, Sort, Update **Description** Indicates that you're using the latest version of the Channel Order App (COA). To ensure that your order is processed, check this field.
Label Contract **Name** `Partner_Contract_Rules__c`	**Type** reference **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** Lookup to the related contract terms record. This field is required.
Label Customer Name **Name** `Customer__c`	**Type** reference **Properties** Create, Filter, Group, Nillable, Sort, Update

Field	Details
	Description Lookup to a customer record. Specify an existing customer record. You can't populate customer details using the API. This field is required.
Label Date Partner Received Customer Order **Name** `Date_Partner_Received_Customer_Order__c`	**Type** date **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** Date you received the order from the customer. This field is required.
Label Date Customer Accepted SFDC Service Agreement **Name** `Date_Customer_Accepted_SFDC_Svc_Agrmnt__c`	**Type** date **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** Date the customer accepted the Salesforce service agreement. This field is required for OEM contracts.
Label Error Comment **Name** `Error_Comment__c`	**Type** textarea **Properties** Create, Nillable, Sort, Update **Description** Stores comments or instructions from Salesforce Partner Operations when a submitted order can't be processed.
Label I Certify a Corresponding Order is Rec'd **Name** `I_certify__c`	**Type** picklist **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** Confirmation that the order was received. Possible values are `Yes` and `No`. This field is required.
Label Order Type **Name** `Order_Type__c`	**Type** picklist **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** The type of order that you're submitting for processing and activation. Possible values are `Initial`, `Add-On`, `Reduction`, `Cancellation Order`, `Upgrade`

Field	Details
	- Partner App, and Upgrade - Org Edition. Specify Upgrade - Partner App for a renewal order. Specify Upgrade - Org Edition for an upgrade order. This field is required.
Label Service Order Status **Name** Service_Order_Status__c	**Type** picklist **Properties** Create, Defaulted on create, Filter, Group, Nillable, Sort, Update **Description** Status of the order. Possible values are Draft, Submitted, Received, In Process, Error, Activated, and Provisioned. You can submit only orders with a status of Draft.
Label Service Start Date **Name** Service_Start_Date__c	**Type** date **Properties** Create, Filter, Group, Sort, Update **Description** Date to activate or provision the customer's order. You can specify today's date or a date in the future. This field is required.

Service Order Detail

Represents an instance of a product on a service order.

Note: Field names are prefixed with CHANNEL_ORDERS__ unless otherwise noted.

When you submit an order with the Channel Order App API, include the following fields.

Fields

Field Name	Details
Label App **Name** Application__c	**Type** string **Properties** Create, Filter, Group, Nillable, Sort **Description** Name of the app associated with the product.
Label Billing Frequency	**Type** double

Field Name	Details
Name `pc_Billing_Frequency__c`	**Properties** Create, Filter, Nillable, Sort, Update **Description** How often the customer is billed per year. This value must match your Salesforce contract, unless you've been granted override permissions.
Label Cancellation Terms (days) **Name** `pc_Cancellation_Terms__c`	**Type** double **Properties** Create, Filter, Nillable, Sort, Update **Description** Number of days the customer has to cancel the contract. This value must match your Salesforce contract, unless you've been granted override permissions.
Label Contract Auto Renew **Name** `pc_Contract_Auto_Renew__c`	**Type** picklist **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** Whether the contract automatically renews at the end of the term. Possible values are `Yes` and `No`. This value must match your Salesforce contract, unless you've been granted override permissions.
Label Contract Length **Name** `pc_Contract_Length__c`	**Type** double **Properties** Create, Filter, Nillable, Sort, Update **Description** Length of the contract in months. This value must match your Salesforce contract, unless you've been granted override permissions.
Label Currency **Name** `Currency__c`	**Type** string **Properties** Filter, Nillable, Sort **Description** The default contract currency from the contract terms associated with this order. Read-only.
Label Customer Price **Name** `Customer_Price_Per_Month__c`	**Type** double **Properties** Create, Filter, Nillable, Sort, Update

Field Name	Details
	Description Price per unit per month. This field is required for PNR products.
Label Fixed Price **Name** `pc_Fixed_Price__c`	**Type** double **Properties** Create, Filter, Nillable, Sort, Update **Description** Fixed price of the product at the time the order was created. This field must be explicitly set when using the API.
Label Partner Contract Term **Name** `pc_Partner_Contract_Term__c`	**Type** reference **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** Lookup to the related contract terms record.
Label PNR % **Name** `pc_PNR__c`	**Type** double **Properties** Create, Filter, Nillable, Sort, Update **Description** Percent net revenue of the product at the time the order was created. This field must be explicitly set when using the API.
Label Pricing **Name** `pc_Pricing_Type__c`	**Type** picklist **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** Pricing model of the product. Possible values are `Fixed` and `PNR`. This field must be explicitly set when using the API.
Label Product **Name** `Product_Name__c`	**Type** reference **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** Lookup to the related product catalog record.

Field Name	Details
Label Product ID **Name** `pc_Product_ID__c`	**Type** string **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** ID of the product. This field must be explicitly set when using the API.
Label Renewal Terms (months) **Name** `pc_Renewal_Terms__c`	**Type** double **Properties** Create, Filter, Nillable, Sort, Update **Description** Renewal term in months. This value must match your Salesforce contract, unless you've been granted override permissions.
Label Service Order **Name** `Partner_Order__c`	**Type** reference **Properties** Create, Filter, Group, Sort **Description** Lookup to the related service order record.
Label SFDC Invoice Description **Name** `Product_Line_Description__c`	**Type** string **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** Contains additional invoice details for the product or order. This field is optional.
Label Total Quantity **Name** `Quantity__c`	**Type** double **Properties** Create, Filter, Nillable, Sort, Update **Description** Number of product catalogs on the service order.

Partner Order Submit API

(No longer supported and available only in version 1.39 and earlier of the Channel Order App.) Send orders to Salesforce immediately or asynchronously using the Partner Order Submit API.

Important: In Channel Order App (COA) v2.0 and later, the Channel Order Apex API replaces the Partner Order Submit API. If you have any existing integrations with the Partner Order Submit API, migrate them to the Channel Order Apex API.

Syntax

```
channel_orders.ServiceOrderProcessor.sendOrder()
channel_orders.ServiceOrderProcessor.sendOrderAsync()
```

Note: When you submit an order using `sendOrder` or `sendOrderAsync`, include an order ID or set of order IDs as the argument. For example, `channel_orders.ServiceOrderProcessor.sendOrder(orderId)`.

Usage

Use `sendOrderAsync` when you want to create or update multiple orders and send them in the same transaction. See the example in this section for more details.

Rules and Guidelines

This is an Apex implementation, so all Apex usage rules and limits apply. Salesforce supports only one order per call.

Use the Partner Submit API to send an order after it has been created using a valid Service Order ID. You can create Service Order and Service Order Detail records using the Channel Order App, data loading, or automated processing.

Each order must include the fields listed on the Service Order and Service Order Detail objects.

Methods

The ServiceOrderProcessor object supports the following methods.

Name	Arguments	Description
`sendOrder`	ID	Submit an order with a single ID immediately.
`sendOrder`	Set of IDs	Submit an order with a set of IDs immediately.
`sendOrderAsync`	ID	Submit an order with a single ID asynchronously (`@future`).
`sendOrderAsync`	Set of IDs	Submit an order with a set of IDs asynchronously (`@future`).

Example: Batching on the Partner Order Submit API

You can only invoke ServiceOrderProcessor once per Apex transaction. If you pass a set of IDs to `sendOrder` or `sendOrderAsync`, the maximum set size is 5. This example uses a batch job to work around this limitation.

In this example, if you have 100 orders in Draft status, the code creates one batch job with 100 executions, because only one record is processed per execution.

```
//Batch Apex class
global class COABatchClass implements Database.batchable<sObject>, Database.AllowsCallouts,
 Database.Stateful{
   final String DRAFT_STATUS = 'Draft';
```

```
   global final String query =
      'select Id, CHANNEL_ORDERS__Service_Order_Status__c ' +
     ' from CHANNEL_ORDERS__Service_Order__c where CHANNEL_ORDERS__Service_Order_Status__c
 =: DRAFT_STATUS';

   global Database.QueryLocator start(Database.BatchableContext BC){
      return Database.getQueryLocator(query);
   }

  global void execute(Database.BatchableContext info, List<CHANNEL_ORDERS__Service_Order__c>
 scope){
       for(CHANNEL_ORDERS__Service_Order__c s : scope){
         CHANNEL_ORDERS.ServiceOrderProcessor.sendOrder(s.Id);
         }
   }
   global void finish(Database.BatchableContext BC){}
}

//Batch call
Id batchInstanceId = Database.executeBatch(new COABatchClass(), 1);
```

CHAPTER 9 Manage Licenses

In this chapter ...

- License Management App
- Get Started with the License Management App
- Manage Leads and Licenses for Your Offering
- Troubleshoot the License Management App
- License Management App FAQ

Learn how to use the License Management App (LMA) to manage leads and licenses for your AppExchange solutions, and how to provide administrative support for your customers.

License Management App

The License Management App (LMA) lets you manage leads and licenses for your AppExchange offerings. By integrating the LMA into your sales and marketing processes, you can better engage with prospects, retain existing customers, and grow your ISV business.

Note: The LMA is available in English only.

The LMA is available to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, visit https://partners.salesforce.com.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

How Does the License Management App Work?

Each time a customer installs your packaged offering, the License Management App (LMA) creates lead and license records. To use the LMA effectively, it's important to understand how that process works.

Integrate the License Management App into Your Business Processes

Our most successful partners don't just use the License Management App (LMA) to manage leads and licenses. Instead, they integrate the LMA into their existing business processes and with other Salesforce tools. Here are some examples of how you can use the LMA to grow your business and retain customers.

Best Practices for the License Management App

Follow these guidelines and best practices when you use the License Management App (LMA).

How Does the License Management App Work?

Each time a customer installs your packaged offering, the License Management App (LMA) creates lead and license records. To use the LMA effectively, it's important to understand how that process works.

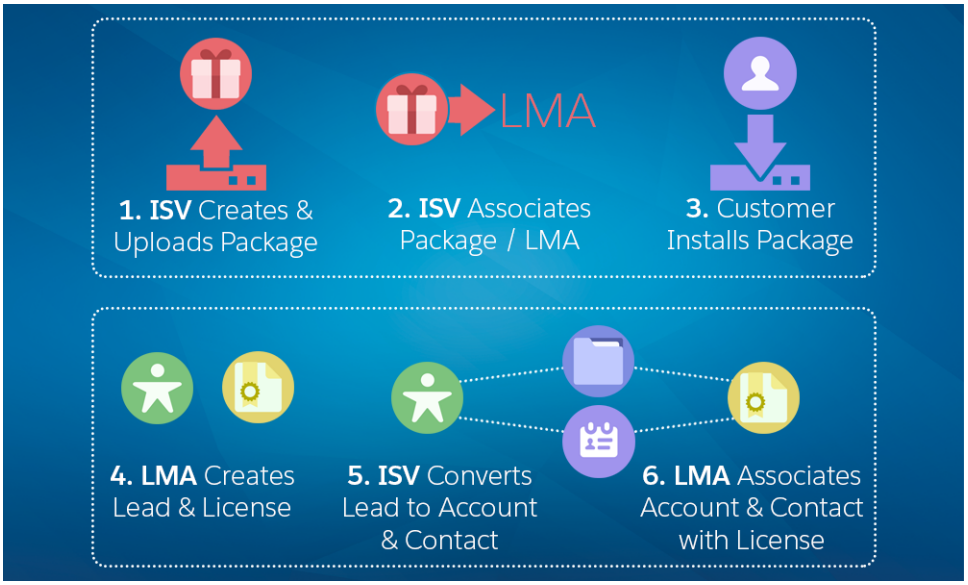
Packages, Leads, and Licenses

The key objects in the LMA are packages, leads, and licenses.

- *Packages* are container for apps or Lightning components and can be either managed or unmanaged. In the LMA, packages refer to managed packages that have been uploaded to AppExchange. Packages can have one or more versions, and each package version can have multiple licenses.
- *Leads* give you details about who installed your offering, such as the installer's name, company, and email address. Leads generated by the LMA are just like the ones you use elsewhere in Salesforce, except the lead source is Package Installation. When you set up the LMA, you designate a *lead manager* in your org to receive the new leads. You can manually convert leads into accounts and contacts in the LMA. The license is then associated with the converted account and contact.
- *Licenses* give you control over how many users in the customer's org can access your offering and for how long. Licenses are unique to the LMA, and each license has a direct relationship with a lead.

How Leads and Licenses Are Created

Lead and license records are the end result of a process that involves, you, the customer, and the LMA. Here's how the process unfolds, starting with the creation of your package.



Step	Who Does This	Where This Happens
[1] Your offering is packaged, and a version is uploaded to the AppExchange.	You (ISV partner)	Your Developer Edition org
[2] Your package version is associated with the LMA, and default license values are set.	You (ISV partner)	The Partner Community
[3] Your offering is installed as part of a purchase or trial.	Customer or prospect	Any compatible org
[4] A lead record is created with the customer's name, company, and email address.	The LMA	Your business org
[4] License records are created according to the default values you specified.	The LMA	Your business org The customer's org
[5] The lead record is converted to account and contact records.	You (ISV partner)	Your business org
[6] Account and contact records are associated with the license record.	The LMA	Your business org

LMA Packages

In the LMA, packages refer to managed packages that have been uploaded to the AppExchange for distribution. Packages can have one or more versions, and each package version can have multiple licenses. Package version has a master-detail relationship with a package. The package object is the root object for all information in the LMA.

LMA Licenses

Licenses give you control over how many users in the customer's org can access your offering and for how long. Licenses are unique to the LMA, and each license has a direct relationship with a lead. Licenses have lookup relationships with leads and package versions.

LMA Packages

In the LMA, packages refer to managed packages that have been uploaded to the AppExchange for distribution. Packages can have one or more versions, and each package version can have multiple licenses. Package version has a master-detail relationship with a package. The package object is the root object for all information in the LMA.

In the LMA, from **Packages**, select a package name to view its details, including information about the org where you developed it. In the Package Version related list, you can see all the uploaded and registered package versions on the AppExchange

Important: Don't edit, delete, clone, or create packages, package versions, or licenses. These records are automatically created and contain important information for tracking the licenses and packages in the License Management App. They can't be repopulated.

Package Details

A package contains the following information.

Field	Description
`Created By`	Defaults to the License Manager.
`Developer Name`	The name of the org where you developed the package.
`Developer Org ID`	The 18-character ID of the org where you developed the package.
`Last Modified By`	The name of the last user to modify this record, along with the date and time it was updated.
`Latest Version`	The most recent uploaded and registered version of the package. You enter this information when uploading the package.
`Lead Manager`	The owner of leads created when customers install your package. `Lead Manager` is blank when the package record is created. If you don't assign a lead manager, the License Management App owns the lead.
`Owner`	The License Management App. Don't change this value.
`Package ID`	The 18-character GUID (Globally Unique ID) that identifies the package.
`Package Name`	The name you specified when you created the package.
`Release Date`	The date you uploaded this package to the AppExchange.

Package Version Details

A package version contains the following information.

Field	Description
`Beta`	Indicates an early version of a managed package for testing by your customers. You specify beta status when you upload the package to the AppExchange
`Created By`	Defaults to the License Management App.
`Last Modified By`	The name of the last user to modify this record, along with the date and time it was updated.
`Package`	The package for which this is a package version.
`Package Version Name`	The name you specified when you created the package.
`Release Date`	The date you uploaded this package to the AppExchange.
`Version`	The version, as specified during upload to the AppExchange.
`Version ID`	The 18-character ID of this package version.

LMA Licenses

Licenses give you control over how many users in the customer's org can access your offering and for how long. Licenses are unique to the LMA, and each license has a direct relationship with a lead. Licenses have lookup relationships with leads and package versions.

In the LMA, from **Licenses**, select a license record to view details including status, package version, owner, and install date.

Important: Don't edit, delete, clone, or create packages, package versions, or licenses. These records are automatically created and contain important information for tracking the licenses and packages in the License Management App. They can't be repopulated.

License Details

A license contains the following information.

Field	Description
`Account`	The account for a converted lead.
`Contact`	The contact for a converted lead.
`Created By`	Defaults to the License Manager.
`Expiration Date`	Displays the expiration date or `Does not expire` if the license does not expire. The default is `Does not expire`.
`Information Current As Of`	The last time Salesforce retrieved information about the installer's org.
`Install Date`	The date the customer installed this package version.
`Instance`	The Salesforce instance where the installer's org resides.
`Last Modified By`	The name of the last user to modify this record, along with the date and time it was updated.
`Lead`	The lead that the LMA created when the app was installed. A lead represents the user who owns the license. If you convert the lead into an opportunity, the lead name is retained but the lead record no longer exists. If you click the link, a page states that the lead has been converted.
`License Name`	A number that represents an instance of a license. The number is incremented by one for each new license.
`Licensed Seats`	Displays the number of licenses or `Site License`. The default is `Site License`.
`License Status`	Indicates the type of license. Available values are Trial, Active, Suspended, and Uninstalled.
`License Type`	Indicates whether the license is editable.
`Org Edition`	The edition of the installer's org.
`Org Expiration Date`	If the installer is using a trial org, the date when the trial expires.
`Org Status`	The status of the installer's org. Possible values include Trial or Active.
`Owner`	Always the License Management App. Don't change this value.
`Package Version`	Links to the package version that is the parent of this license.
`Package Version Number`	The version number of the installed package.

Field	Description
`Sandbox`	Indicates whether the license is for a package installed in a sandbox org.
`Subscriber Org ID`	A globally unique 15-character ID representing the installer's org.
`Used Licenses`	Displays the number of users who have a license to the package. This field is blank if: • A customer uninstalled the package. • `Licensed Seats` is set to Site License.

Limits

You can allocate up to 99,000,000 seats per subscriber license.

Integrate the License Management App into Your Business Processes

Our most successful partners don't just use the License Management App (LMA) to manage leads and licenses. Instead, they integrate the LMA into their existing business processes and with other Salesforce tools. Here are some examples of how you can use the LMA to grow your business and retain customers.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Alert Sales Reps Before a License Expires

If you're managing licenses for several offerings, it can be difficult to keep track of what expires and when. If a license expires accidentally, you could even lose a customer. To help your customers with renewals, set up a workflow rule to email a sales rep on your team before the license expires.

To automatically email the sales rep, follow these high level steps.

1. Create an email template for the notification.
2. Create a workflow rule with a filter that specifies enough time before the expiration date to discuss renewal options.
3. Associate the workflow rule with a workflow alert that sends an email to the appropriate team member or sales rep.

Notify Customer-Retention Specialists When an Offering Is Uninstalled

If a customer uninstalls your offering, find out why. By speaking to the customer, you have an opportunity to restore the business relationship or receive feedback that helps you improve your offering.

To notify a customer-retention specialist on your team,follow these high level steps.

1. Create an email template for the notification.
2. Create a workflow rule with a filter that specifies that the `License Status` equals *`Uninstalled`*.
3. Associate the workflow rule with a workflow alert that sends an email to the retention specialist.

Best Practices for the License Management App

Follow these guidelines and best practices when you use the License Management App (LMA).

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

- Set up My Domain in the Salesforce org where the LMA is installed. A custom domain prevents you from being logged out of your org when you use the Subscriber Support Console to help customers troubleshoot issues. For more information, see "My Domain" in the Salesforce online help.
- Create a list view filter for leads created by installed packages. The filter helps your team separate subscriber-based leads from leads coming from other sources.
- Use the API to find licensed users. The `isCurrentUserLicensed` method determines if a user has a license to a managed package. For more information, see the *Lightning Platform Apex Code Developer's Guide*.
- Don't create workflow rules, triggers, or validation rules that require custom fields on the license or lead objects. Likewise, don't impose conditions on updating or creating license or lead records. These kinds of customizations prevent the LMA from working.
- Don't create required custom fields on lead, license, package and package version objects.
- Don't define `before-create` triggers or validation rules on lead, license, package, or package version objects.

Get Started with the License Management App

To start managing leads and licenses with the License Management App (LMA), install the LMA in your Salesforce org. Then associate at least one package that you've uploaded to the AppExchange and configure the LMA.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Install the License Management App

Install the License Management App (LMA) in the production Salesforce environment where you manage sales, billing, and marketing at your company. If you received a Partner Business Org when you joined the Partner Program, the LMA is pre-installed there, so you can skip this step.

Associate a Package with the License Management App

To receive lead and license records for an offering, you associate a package with the Salesforce org in which the License Management App (LMA) is installed. Before you associate a package with the LMA, upload the package to the AppExchange. You can only manage licenses for managed packages.

Configure the License Management App

After you associate a managed package with the LMA, assign a lead manager and set object permissions so that people on your team can use the LMA.

Install the License Management App

Install the License Management App (LMA) in the production Salesforce environment where you manage sales, billing, and marketing at your company. If you received a Partner Business Org when you joined the Partner Program, the LMA is pre-installed there, so you can skip this step.

USER PERMISSIONS

To install packages:

- Download AppExchange Packages

Important: Don't install the LMA in the Developer Edition org where you created a managed package.

1. Log a case in the Partner Community requesting the LMA.

   a. Log in to the Partner Community and go to the Support tab.
   b. Select **New Case**.
   c. Select **License Management Application**, and then select **Create a Case**.
   d. For Subtopic, select **Request LMA**.
   e. Enter the required information in the `Description` field, and then select **Submit Case**.
      After Partner Operations reviews the case, you receive an email with an installation URL.
2. Log in to the org where you want to install the LMA, and then select the installation URL.
3. Choose which users can access the LMA, and then select **Install**.
4. Confirm that you have installed the LMA by opening the app launcher in Lightning Experience or app menu in Salesforce Classic. If the installation was successful, License Management App appears in the list of available apps.

Associate a Package with the License Management App

To receive lead and license records for an offering, you associate a package with the Salesforce org in which the License Management App (LMA) is installed. Before you associate a package with the LMA, upload the package to the AppExchange. You can only manage licenses for managed packages.

USER PERMISSIONS

To manage licenses in the Partner Community:
- Manage Listings

1. Log in to the Partner Community and go to the Publishing tab.
2. Note: If you've already linked your packaging org, you can skip this step.

   Connect your packaging org to the publishing console.
   a. On the Organizations tab, select **Connect Organization**.
   b. Enter the login credentials for the org in which you created the package, and then select **Submit**.
3. Associate your package with the LMA.
   a. On the Packages tab, select the package you want to associate with the LMA, and then select **Manage Licenses**.
   b. Select **Register**.
   c. Enter the login credentials for the org in which the LMA is installed, and then select **Submit**.
   d. Choose default license values for your offering, and then select **Save**.
      It can take 30 minutes or more to associate a package record with the LMA.

You associate a managed package with the LMA only once. After a package is associated, the new versions that you create are automatically linked to the LMA.

Configure the License Management App

After you associate a managed package with the LMA, assign a lead manager and set object permissions so that people on your team can use the LMA.

USER PERMISSIONS

To configure the LMA:
- System Administrator profile

To edit licenses and packages:
- Read

  AND

  Edit

To view licenses, packages, and package versions:
- Read

1. Assign a lead manager. If you don't assign a lead manager, you don't receive the lead records that are created when customers install your offering.
   a. Select a package in the LMA.
   b. Select **Edit**.
   c. For Lead Manager, search for a user. In most cases, the lead manager is someone from your sales team.
   d. Select **Save**.
2. Set custom object permissions.

   Note: Users with the System Administrator profile can create, modify, and delete these objects by default because they have the "Modify All Data" permission.

Licenses	Most users in your org don't need any permissions. Users who view licenses need the "Read" permission. Users who modify license records need "Read" and "Edit" permissions.
Packages	Only users who assign the lead manager need "Edit" permission. Other users have either "Read" permission or no permissions.
Package Versions	All users have "Read" permission or no permissions, because they don't need to create, modify, or delete these records.

3. Set field-level security in user profiles or permission sets.

Licenses	Your settings depend on how you want to manage these fields for different users in your org.
Packages	Make all fields Read-Only.
Package Versions	Make all fields Read-Only.

4. To use the `Modify License` Visualforce page, override the Edit control on the license record.
5. Add related lists.
   - Add the Licenses related list to the appropriate Lead page layouts. License managers can use this list to view the licenses associated with a particular lead.
   - Add the Licenses related list to the appropriate Account page layouts. Users can view this list and identify the licenses associated with a particular account.
   - Add the Licenses related list to the appropriate Contact page layouts. Users can view this list and identify the licenses associated with a particular contact.

Manage Leads and Licenses for Your Offering

After you configure the LMA, you can change lead manager, modify license records, and refresh licenses.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Modify a License Record in the License Management App

You can change a customer's access to your offering by modifying a license record. For example, increase or decrease the number of seats included with a license or change the expiration date.

Change the Lead Manager in the License Management App

You can change who receives leads created when a customer or prospect installs your offering from the AppExchange. Usually, the lead manager is someone from your sales team. When new leads are created in the License Management App (LMA), the `Lead Owner` field on lead records defaults to the package's lead manager. If you haven't specified a lead manager, the lead owner defaults to the LMA.

Refresh Licenses for an Offering in the License Management App

Refresh licenses to sync license records for a package across all customer installations. Consider refreshing licenses if discrepancies appear between the number of licenses in a customer's org and the License Management App (LMA) or if you installed the LMA in a new org.

Move the License Management App to Another Salesforce Org

By default, the License Management App (LMA) is installed in your Partner Business Org (PBO). Salesforce strongly recommends managing licenses from your PBO. However, if your company chooses to use another org for ISV business processes, you can install the LMA in that org.

Modify a License Record in the License Management App

You can change a customer's access to your offering by modifying a license record. For example, increase or decrease the number of seats included with a license or change the expiration date.

USER PERMISSIONS

To edit licenses and packages:

- Read

  AND

  Edit

1. Go to a license record in the License Management App (LMA).
2. Select **Modify License**. If you don't see Modify License, edit the page layout to add the control.

   Warning: In Salesforce Classic and Lightning Experience, sometimes the license detail page includes an Edit control. Don't use this control—use **Modify License** instead.

3. Update field values as needed.

Field	Description
`Expiration`	Enter the last day that the customer can access your offering, or select **Does not expire** if the license doesn't expire.
`Seats`	Enter the number of licensed seats, or select **Site License** to make your offering available to all users in the customer's org. You can allocate up to 99,000,000 seats.
`Status`	Select a value from the drop-down list. • **Trial**—Allows the customer to try your offering for up to 90 days. After the trial license converts to an active license, it cannot return to a trial state. • **Active**—Allows the customer to use your offering according to the license agreement.

Field	Description
	• **Suspended**—Prohibits the customer from accessing your offering. Note: When your offering is uninstalled, its status is set to Uninstalled, and the license can't be edited.

4. Select **Save**.

Change the Lead Manager in the License Management App

You can change who receives leads created when a customer or prospect installs your offering from the AppExchange. Usually, the lead manager is someone from your sales team. When new leads are created in the License Management App (LMA), the `Lead Owner` field on lead records defaults to the package's lead manager. If you haven't specified a lead manager, the lead owner defaults to the LMA.

1. Go to a package in the LMA. If you don't see any packages, check your list view.
2. Select **Edit**, and then locate a user. Make sure that you're selecting someone who has permission to access license records in the LMA.
3. Select **Save**.

USER PERMISSIONS

To edit licenses and packages:
- Read

  AND

  Edit

Refresh Licenses for an Offering in the License Management App

Refresh licenses to sync license records for a package across all customer installations. Consider refreshing licenses if discrepancies appear between the number of licenses in a customer's org and the License Management App (LMA) or if you installed the LMA in a new org.

Note: You can refresh licenses for a package once per week.

1. Go to a package record in the LMA. If you don't see any packages, check your list view.
2. Select **Refresh Licenses**. In Lightning Experience, you might need to select the drop-down list to see this control.
3. Confirm that you want to refresh licenses for this package, and then select **Refresh Licenses** again.

USER PERMISSIONS

To edit licenses and packages:
- Read

  AND

  Edit

Move the License Management App to Another Salesforce Org

By default, the License Management App (LMA) is installed in your Partner Business Org (PBO). Salesforce strongly recommends managing licenses from your PBO. However, if your company chooses to use another org for ISV business processes, you can install the LMA in that org.

Important: When you move the LMA to a new org, you must manually re-associate your packages and refresh the licenses. Your package and license records don't move to the new org.

1. Log a case to break the association between the LMA and the org where it's currently installed.
   a. Log in to the Partner Community and go to the Support tab.
   b. Select **New Case**.
   c. Select **License Management Application**, and then select **Create a Case**.
   d. For Subtopic, select **Other**.
   e. Enter the required information in the `Description` field, and then select **Submit Case**.
2. Install the LMA in the new org on page 232.
3. Associate your packages with the new org on page 233.
4. Refresh licenses for your packages on page 236.

USER PERMISSIONS

To install packages:
- Download AppExchange Packages

To manage licenses in the Partner Community:
- Manage Listings

To edit licenses and packages:
- Read

  AND

  Edit

Troubleshoot the License Management App

The most frequent problems arise when leads and licenses aren't created or a proxy user is deactivated.

Leads and Licenses Aren't Being Created

When a customer installs your package, leads or licenses are created. If they aren't, check the configuration in the org in which the LMA is installed. If you resolve the issue with one of these recommendations, the licenses usually appear in the LMA after a few days.

Proxy User Has Deactivated Message

If a "proxy user has deactivated" message appears when editing a license in the LMA, a subscriber org could be locked, deleted, or disabled. Here's a list of things to check.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Leads and Licenses Aren't Being Created

When a customer installs your package, leads or licenses are created. If they aren't, check the configuration in the org in which the LMA is installed. If you resolve the issue with one of these recommendations, the licenses usually appear in the LMA after a few days.

Did the customer really install the package?

When a customer selects **Get it Now** on your AppExchange listing, Salesforce counts this as an installation. However, the customer can cancel the installation before it completed, or the installation could have failed. If the installation doesn't finish, a license doesn't appear in the LMA.

Is State and Country picklist validation enabled?

If it is enabled, try disabling it. A known issue prevents leads from being created in the LMA if this feature is enabled. The issue occurs if customers haven't provided state and country values in their user profiles, or those values are incorrect.

Does the lead or license object have a trigger?

Don't use `before_create` or `before_update` triggers on leads and licenses in the LMA. Instead, use `after_` triggers, or remove all triggers. If a trigger fails, it can block license creation.

Does the lead or license record have a required custom field?

If yes, remove the requirement. The LMA doesn't populate required custom field, so it can prevent licenses or leads from being created.

Is the lead manager a valid, active user?

If not, the LMA can't create leads and licenses.

Does the lead or license record have a validation rule?

Validation rules often block the creation of LMA lead or license records because the required field isn't there.

Does the lead or license have a workflow rule?

Workflow rules sometimes prevent leads and licenses from being created. Remove the workflow rule.

Was the lead converted to an account?

When leads are converted to accounts, they are no longer leads.

Proxy User Has Deactivated Message

If a "proxy user has deactivated" message appears when editing a license in the LMA, a subscriber org could be locked, deleted, or disabled. Here's a list of things to check.

Is the org active?

Check to see if the customer deleted the org. If the org has been deleted, delete the corresponding license record.

Has the package been installed?

If the org is locked or the package has been uninstalled, the license record can't be updated. Ask the customer to reinstall the package.

License Management App FAQ

Answers to common questions about the License Management App (LMA).

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Is the LMA compatible with Lightning Experience?

Can I install the LMA in a non-production Salesforce org?

Why can't I see the Modify License button on my license records?

A customer installed my package before I associated it with the LMA. How can I manage the license record?

Can I automate the assignment of licenses to users in the subscriber org?

Why aren't leads and licenses being created in the LMA?

What happens when I decrease the number of available licenses below the current number of licensed users?

Is the LMA compatible with Lightning Experience?

Yes, both Salesforce Classic and Lightning Experience support the LMA.

Can I install the LMA in a non-production Salesforce org?

By default, the LMA is installed in your Partner Business Org, a production environment that includes the ISV tools needed to run your business. When the LMA is part of a production environment, you can fully integrate license management into your sales, billing, and marketing processes. Our most successful partners make use of the LMA in their production orgs.

Additionally, some types of non-production environments, such as trial orgs, eventually expire. If this happens, your mission critical data becomes temporarily inaccessible. For these reasons, Salesforce strongly discourages moving the LMA to a non-production org.

Why can't I see the **Modify License** button on my license records?

You must add the Modify License button to the layout of your license detail page.

A customer installed my package before I associated it with the LMA. How can I manage the license record?

Go to the package's detail page in the LMA, and then select **Refresh Licenses**. A license record for this customer now appears.

Can I automate the assignment of licenses to users in the subscriber org?

Yes, you can use the API to assign or revoke licenses for managed packages installed in your Salesforce org. For more information, see the PackageLicense and UserPackageLicense objects in the *SOAP API Developer Guide*.

Why aren't leads and licenses being created in the LMA?

Common reasons why leads and licenses aren't created in the LMA include:

- You haven't associated the package or package version with the LMA.
- Lead, license, package, or package version custom objects have mandatory custom fields. Try removing the mandatory custom fields.
- The lead manager is not a valid, active user.
- `before_` triggers are preventing lead creation. Try removing the triggers, or use `after_` triggers instead.

What happens when I decrease the number of available licenses below the current number of licensed users?

Users in the customer's org who already have access to your offering continue to have access. Their system administrator must manually revoke the extra licenses. Until the admin revokes access, the license count remains negative.

CHAPTER 10 Manage Features

In this chapter ...

- Feature Parameter Metadata Types and Custom Objects
- Set Up Feature Parameters
- Reference Feature Parameters to Drive App Behavior and Track Activation Metrics
- Hide Custom Objects and Custom Permissions in Your Subscribers' Orgs
- Best Practices for Feature Management
- Considerations for Feature Management

Take the License Management App (LMA) a step further by extending it with the Feature Management App (FMA). The FMA is generally available as of mid-October 2017.

Here at Salesforce, we sometimes run pilot programs, like the one we ran when we introduced Feature Management. Sometimes we dark-launch features to see how they work in production before sharing them with you. Sometimes we make features available to select orgs for limited-time trials. And sometimes we want to track activation metrics for those features.

With feature parameters, we're extending this previously secret functionality to you, our Partner Ohana. Install the FMA in your License Management Org (LMO). The FMA extends the License Management App, and like the LMA, it's distributed as a managed package.

To try out Feature Management in a sample Salesforce DX project, clone our Project Force App on GitHub.

Feature Parameter Metadata Types and Custom Objects

Feature parameters are represented as Metadata API types in your packaging org, as records of custom objects in your LMO, and as hidden records of custom objects in your subscriber's org. The FMA creates the custom objects. Three types of feature parameters store three types of values: boolean, integer, and date. You can reference these values in your code, just like you reference any other value in a customer's org.

Feature Parameter Fields

Feature parameters are represented as Metadata API types that you can work with in your packaging org. The `FeatureParameterBoolean`, `FeatureParameterDate`, and `FeatureParameterInteger` types store three types of values: boolean, integer, and date. You can use these types in managed packages to store these values:

- A `dataFlowDirection` value: `LmoToSubscriber` or `SubscriberToLmo`.
- A `masterLabel` value for each feature parameter.
- A default `value` for each feature parameter. You can reference the values in your code, just like you reference other values in a customer's org.

The first time a subscriber installs your package, a `FeatureParameter__c` record is created in your LMO for each feature parameter. The feature parameter records store values in these fields:

- `FullName__c`
- `DataType__c` (`Boolean`, `Integer`, or `Date`)
- `DataFlowDirection__c`
- `Package__c`
- `IntroducedInPackageVersion__c`
- `Namespace_Prefix__c`

In your LMO and in your subscriber's org, records of custom junction objects represent your feature parameters: records of `FeatureParameterBoolean__c`, `FeatureParameterDate__c`, and `FeatureParameterInteger__c` objects. The FMA creates records of these junction objects in the LMO and in the customer's org when a subscriber installs your package. These records associate your feature parameters with the licenses for your subscribers and set the feature parameters' values. Their values in your LMO and in your subscriber's orgs are linked. Each record stores three values:

- `FeatureParameter__r`
- `License__c`
- `Value__c`

Life Cycle of a Feature Parameter

Let's make all that information about feature parameters and their fields a bit more relevant. Here's a brief overview of how these types and objects work, from start to finish.

1. The ISV defines feature parameters in the packaging org via the Feature Parameters tab on the Package detail page for the org's managed package. Depending on the value of `dataFlowDirection` (`LMO to Subscriber` or `Subscriber to LMO`), the feature parameters alter the associated data in subscriber orgs or collect activation metrics. The ISV then writes other code that interacts with the feature parameters to check access rights or collect usage information.
2. Customers install the package from AppExchange.

3. When the package is installed in a subscriber org, one `FeatureParameter__c` record for each feature parameter is created in the LMO, unless the records already exist.
4. During package installation in a subscriber org, junction object records are created in the LMO and in the subscriber's org. For each feature parameter, a record of a junction object is created in each org to associate the feature parameter with the license for the subscriber org. A junction object is a custom object with two master-detail relationships. In this case, the relationships are with `FeatureParameter__c` and `License__c`. The junction objects are of type `FeatureParameterBoolean__c`, `FeatureParameterDate__c`, or `FeatureParameterInteger__c`. The records store the value of their associated feature parameter for the subscriber org. Initially, their `Value__c` field is populated with the `defaultValue` from the packaging org. Their values in the LMO and in the subscriber org are linked.
5. The ISV uses the junction objects to override the feature parameters' default values or to collect data. Depending on the value of each feature parameter's `DataFlowDirection__c` field, data flows to the subscriber org (from the LMO) or to the LMO (from the subscriber org). That data is stored in the junction object records.

SEE ALSO:

GitHub: Project Force App (sample Salesforce DX project for Feature Management)

Set Up Feature Parameters

You first set up the Feature Management App in your License Management Org. Then you define feature parameters in your packaging org and add them to your package.

Install and Set Up the Feature Management App in Your License Management Org

Install the FMA in your LMO. Then add the Feature Parameters tab to your default view, and adjust your page layout for licenses to display related lists for your feature parameters.

Create Feature Parameters in Your Packaging Org

Create a feature parameter in your packaging org, and set its type, default value, and data flow direction.

Add Feature Parameters to Your Managed Package

After you've created some feature parameters, you can add them to a managed package as components and reference them in your code. Feature parameters aren't available in unmanaged packages.

Install and Set Up the Feature Management App in Your License Management Org

Install the FMA in your LMO. Then add the Feature Parameters tab to your default view, and adjust your page layout for licenses to display related lists for your feature parameters.

1. To request access to the FMA, log a case in the Partner Community. The FMA extends the License Management App, so be sure to install the LMA before requesting access to the FMA.
2. To install the FMA, follow the instructions in your welcome email.
3. Add the Feature Parameters tab to your default view. For details, see "Customize My Tabs" in the Salesforce Help.
4. Update your page layout for licenses.
   a. Navigate to a license record's detail page.
   b. Click **Edit Layout**.

c. In the Related Lists section of the License Page Layout Editor, add these lists.
   - Feature Parameter Booleans
   - Feature Parameter Dates
   - Feature Parameter Integers

d. For each related list, add these columns.
   - Data Flow Direction
   - Feature Parameter Name
   - Full Name
   - Master Label
   - Value

Create Feature Parameters in Your Packaging Org

Create a feature parameter in your packaging org, and set its type, default value, and data flow direction.

1. From Setup, enter `Packages` in the Quick Find box, then select **Packages**.
2. In the Packages section, in the Package Name column, select your managed package.
3. On the Feature Parameters tab, click **New Boolean**, **New Integer**, or **New Date**.
4. Give your feature parameter a developer name that meets the standard criteria for developer names. The name must be unique in your org. It can contain only alphanumeric characters and underscores, and must begin with a letter. It can't include spaces, end with an underscore, nor contain two consecutive underscores.
5. Give the feature parameter a label.
6. Set a default value for the feature parameter. If you're creating a Feature Parameter Boolean, you see only a checkbox for Default Value. If you want your default value to be `true`, select this checkbox.
7. Set a data flow direction. To use this feature parameter to control behavior in your subscriber's org, select **LMO to Subscriber**. To collect activation metrics from your subscriber, select **Subscriber to LMO**.
8. Click **Save**.

Add Feature Parameters to Your Managed Package

After you've created some feature parameters, you can add them to a managed package as components and reference them in your code. Feature parameters aren't available in unmanaged packages.

Complete these steps in your packaging org.

1. From Setup, enter `Packages` in the Quick Find box, then select **Packages**.
2. In the Packages section, in the Package Name column, select your managed package.
3. On the Components tab, click **Add**.
4. From the Component Type dropdown, select **Feature Parameter Boolean**, **Feature Parameter Date**, or **Feature Parameter Integer**.
5. Select your feature parameter, and then click **Add to Package**.

Reference Feature Parameters to Drive App Behavior and Track Activation Metrics

You can reference feature parameters in your code.

How Do Feature Parameters Work?
When a subscriber installs your package, Salesforce creates two junction object records for each feature parameter: one in your LMO, and a hidden one in your subscriber's org. The linked records keep the values of each feature parameter in sync between your LMO and your subscriber's org. This linking is similar to the mechanism that keeps license records in sync between your LMO and your subscriber's org.

Drive App Behavior with LMO-to-Subscriber Feature Parameters
Feature parameters with a Data Flow Direction value of `LMO to Subscriber` are writable at your end and read-only in your subscriber's org. These feature parameters serve as permissions or limits. Use LMO-to-subscriber feature parameters to enable or disable new features or to control how many of a given resource your subscriber can use. Or, enable features for a limited trial period. The sky's the limit. Assign values to LMO-to-subscriber feature parameters by updating junction object records in your LMO, and then check those values in your code.

Track Preferences and Activation Metrics with Subscriber-to-LMO Feature Parameters
Use subscriber-to-LMO feature parameters to track feature activation in your subscriber's org. Parameter values are assigned on the subscriber's end and then sent to your LMO. To collect the values, update the feature parameters in your subscriber's org using Apex code. Check with your legal team before obtaining activation metrics from your customers. Use activation metrics to collect only aggregated data regarding feature activation.

How Do Feature Parameters Work?

When a subscriber installs your package, Salesforce creates two junction object records for each feature parameter: one in your LMO, and a hidden one in your subscriber's org. The linked records keep the values of each feature parameter in sync between your LMO and your subscriber's org. This linking is similar to the mechanism that keeps license records in sync between your LMO and your subscriber's org.

SEE ALSO:

Feature Parameter Metadata Types and Custom Objects

Drive App Behavior with LMO-to-Subscriber Feature Parameters

Feature parameters with a Data Flow Direction value of `LMO to Subscriber` are writable at your end and read-only in your subscriber's org. These feature parameters serve as permissions or limits. Use LMO-to-subscriber feature parameters to enable or disable new features or to control how many of a given resource your subscriber can use. Or, enable features for a limited trial period. The sky's the limit. Assign values to LMO-to-subscriber feature parameters by updating junction object records in your LMO, and then check those values in your code.

Assign Override Values in Your LMO
To override the default value of a feature parameter in a subscriber's org, update the appropriate junction object record in your LMO.

Check LMO-to-Subscriber Values in Your Code
You can reference feature parameters in your code, just like you'd reference any other custom object.

Assign Override Values in Your LMO

To override the default value of a feature parameter in a subscriber's org, update the appropriate junction object record in your LMO.

1. Open the license record for a subscriber's installation of your package.
2. In the related list for Feature Parameter Booleans, Feature Parameter Integers, or Feature Parameter Dates, select the feature parameter whose value you want to update.
3. Click **Edit**.
4. Set a value.
5. Click **Save**.

Check LMO-to-Subscriber Values in Your Code

You can reference feature parameters in your code, just like you'd reference any other custom object.

Use these Apex methods with LMO-to-subscriber feature parameters to check values in your subscriber's org.

- `System.FeatureManagement.checkPackageBooleanValue('YourBooleanFeatureParameter');`
- `System.FeatureManagement.checkPackageDateValue('YourDateFeatureParameter');`
- `System.FeatureManagement.checkPackageIntegerValue('YourIntegerFeatureParameter');`

SEE ALSO:

Apex Developer Guide: FeatureManagement Class

Track Preferences and Activation Metrics with Subscriber-to-LMO Feature Parameters

Use subscriber-to-LMO feature parameters to track feature activation in your subscriber's org. Parameter values are assigned on the subscriber's end and then sent to your LMO. To collect the values, update the feature parameters in your subscriber's org using Apex code. Check with your legal team before obtaining activation metrics from your customers. Use activation metrics to collect only aggregated data regarding feature activation.

- `System.FeatureManagement.setPackageBooleanValue('YourBooleanFeatureParameter', booleanValue);`
- `System.FeatureManagement.setPackageDateValue('YourDateFeatureParameter', datetimeValue);`
- `System.FeatureManagement.setPackageIntegerValue('YourIntegerFeatureParameter', integerValue);`

Warning: The `Value__c` field on subscriber-to-LMO feature parameters is editable in your LMO. But don't change it. The changes don't propagate to your subscriber's org, so your values will be out of sync.

SEE ALSO:

Apex Developer Guide: FeatureManagement Class

Hide Custom Objects and Custom Permissions in Your Subscribers' Orgs

Occasionally, you want to include custom permissions or custom objects in a package but not show them to your subscribers. Check with your company's legal team before releasing hidden functionality, and aggregate results that you collect from unknowing subscribers.

To hide custom objects when creating your package, set the value of their Visibility field to `Protected`.

To hide custom permissions when creating your package, from Setup, enter *`Custom Permissions`* in the Quick Find box. Select **Custom Permissions** > *`Your Custom Permission`* > **Edit**. Enable **Protected Component**, and then click **Save**. After your package is installed, use the `System.FeatureManagement.changeProtection()` Apex method to hide and unhide custom objects and permissions.

Warning: For custom permissions, you can toggle the protected value indefinitely. However, after you've released unprotected objects to subscribers, you can't set the visibility to `Protected`. Be sure to protect custom objects that you want to hide before you release the first package version that contains them.

To hide custom permissions in released packages:

- `System.FeatureManagement.changeProtection('`***`YourCustomPermissionName`***`', 'CustomPermission', 'Protected');`

To unhide custom permissions and custom objects in released packages:

- `System.FeatureManagement.changeProtection('`***`YourCustomPermissionName`***`', 'CustomPermission', 'Unprotected');`
- `System.FeatureManagement.changeProtection('`***`YourCustomObjectName__c`***`', 'CustomObject', 'Unprotected');`

SEE ALSO:

Apex Developer Guide: FeatureManagement Class

Best Practices for Feature Management

We suggest that you follow these best practices when working with feature parameters.

- We strongly recommend that you use this feature set in a test package and a test LMO before using it with your production package. Apply changes to your production package only after fully understanding the product's behavior. To try out Feature Management in a sample Salesforce DX project, clone our Project Force App on GitHub.
- Limit the number of feature parameters in your package. Each package can include up to 25 feature parameters.
- Create LMO-to-subscriber feature parameters to enable features from your LMO for individual subscriber orgs. Don't use the Apex code in your managed package to modify LMO-to-subscriber feature parameters' values in subscriber orgs. You can't send the modified values back to your LMO, and your records will be out of sync.

  Use LMO-to-subscriber feature parameters as read-only fields to manage app behavior. For example, use LMO-to-subscriber feature parameters to track the maximum number of permitted e-signatures or to make enhanced reporting available.

- Create subscriber-to-LMO feature parameters to manage activation metrics. Set these feature parameters' values in subscriber orgs using the Apex code in your managed package. For example, use subscriber-to-LMO feature parameters to track the number of e-signatures consumed or to check whether a customer has activated enhanced reporting.

Considerations for Feature Management

Keep these considerations and known issues in mind when working with feature parameters.

- Technically, feature parameter records are creatable and editable in the LMO. However, don't create or modify them—we're keeping them updated for you. Likewise, although the records of subscriber-to-LMO feature parameters' junction objects are editable in the LMO, don't edit them. Ignore these feature parameter–related buttons in your LMO: New, Clone, Submit for Approval, and Save & New.
- We are still making improvements to the UI, and we have in-progress fit-and-finish work planned across the feature set.
- In the FMA, the Introduced in Package Version field on the `FeatureParameter__c` object displays cryptic information. We'll fix this issue in the future.
- When you publish a push upgrade to your managed package, feature parameters in your LMO and your subscribers' orgs are updated asynchronously. Creating and updating the junction object records can take several minutes.
- When you update LMO-to-subscriber values in your LMO, the values in your subscribers' orgs are updated asynchronously. This process can take several minutes.
- When the Apex code in your package updates subscriber-to-LMO values in your subscriber's org, the changes can take up to 24 hours to reach your LMO.

CHAPTER 11 Provide a Free Trial of Your Solution

In this chapter ...

- Why Use Trialforce?
- Trialforce
- Set Up Trialforce
- Provide a Free Trial on the AppExchange
- Provide Free Trials on Your Website
- Modify the Trial for an Upgrade
- Trialforce Best Practices
- Trialforce FAQ

Maximize customer adoption by offering free trials of your solution on AppExchange. Explore the types of trials available and determine the best type for your solution.

Note: This feature is available to eligible partners. For more information on the Partner Program, including eligibility requirements, visit https://partners.salesforce.com.

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.

Trialforce

A Trialforce setup consists of several parts. It's important to understand these parts and the relationship between them before you start using Trialforce.

Trialforce Management Organization (TMO)

The TMO is the starting point for setting up Trialforce and the central location for managing Trialforce after it's set up. You must file a case in the Partner Community to receive your TMO. The two tasks you perform in the TMO are create Trialforce source organizations and define templates for custom branding.

Trialforce Source Organization (TSO)

You use the TSO to create a template for the trial orgs received by your customers. You create the TSO from your TMO. The tasks you perform in a TSO are: install your offering, along with any sample data; specify branding by choosing from the templates you created previously in the TMO; configure the TSO to be exactly as you want your customers to experience it; and generate a Trialforce template, which becomes the basis for all trial organizations.

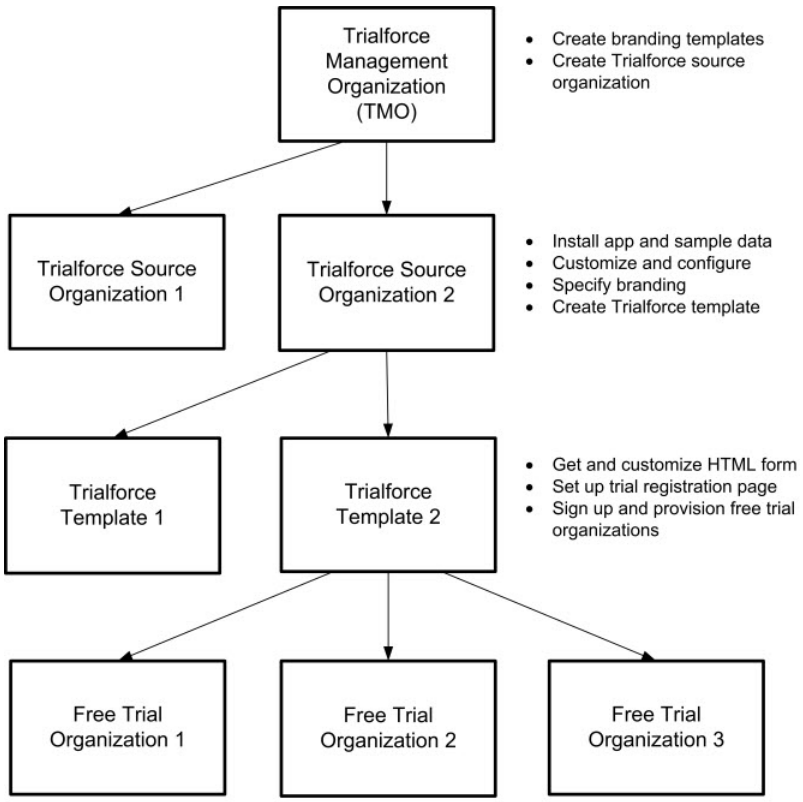
Trialforce Template

The template is a snapshot or exact copy of your TSO at a specific instance in time. You create it from a TSO after you've installed your offering and made configuration changes. The Trialforce template is specified in the HTML page from which customers sign up for trials. It defines the trial organization that is provisioned each time a customer signs up.

HTML Signup Form

This HTML form serves as the registration page on your website from which customers sign up for trials. You must file a case in the Partner Community to get this form and then customize it with your company details. It is associated with the Trialforce template you plan to use for trials. This ensures that each time a customer signs up for a trial on your website, they receive an organization that is an exact copy of your chosen Trialforce template.

Relationship between organizations used to set up Trialforce



The TMO, TSOs, and Trialforce templates have a hierarchical relationship, as illustrated above.

- You can create multiple TSOs from a given TMO. For example, if you want to offer trials for two different apps, you would generate two different TSOs from the same TMO, one for each app. This enables you to use the TMO as a central hub to manage the trials for all Lightning Platform apps or components produced by your company.
- You can create multiple Trialforce templates from the same TSO. For example, if you release a new version of your component after you've started using Trialforce, you can install the upgraded version into the previous TSO and generate a new Trialforce template from it. If you then update your HTML signup form to point to the new Trialforce template, all trial organizations created subsequently have the new version of the package.

As a best practice, we recommend that you have one unique TMO for your company, one TSO for each app or component, and one Trialforce template for each version or edition. Splitting up the configuration process across these different levels makes it easier to maintain and update your trials. Then each time you change something, such as the version, its branding, or a configuration detail of the trial organization, you only need to make the change at one level in the hierarchy. This minimizes the configuration steps involved and makes it easy to concurrently manage trials for multiple products, versions, and editions.

After you've configured a TMO, TSO, and Trialforce template, choose how to provide trials to prospective customers:

- **Using the AppExchange—** Customers begin a trial of your offering directly from an AppExchange listing. This approach is ideal if you're looking for the quickest, easiest way to make a trial available because it requires only a few steps to configure.
- **Using an HTML signup form—** Customers begin a trial of your offering after filling in a customizable HTML signup form. Because you can modify the form's look and feel to match your own website, this approach is ideal for integrating the signup process into your company's web presence.
- **Using the API—** You provision a trial of your app or component programmatically using the SignupRequest API. This approach is ideal if you're looking to have full control of the signup process because it allows for advanced customization.

Set Up Trialforce

After you've built your offering and passed the AppExchange security review, follow these steps to set up Trialforce.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To manage Trialforce:
- Customize Application

Note: To enable Trialforce, you must first sign the ISVforce/OEM agreement.

1. Create your managed package.
2. Configure a License Management Organization (LMO) to manage customers' access to apps and components. If you're an existing Salesforce user, install the License Management Application (LMA) in your CRM organization (Enterprise Edition is required). If you're new to the Partner Program, the LMA is preinstalled in your partner business org.
3. Link a version with the LMO and set the license defaults. This step ensures that each time a prospect creates a trial, the LMO receives a new lead and license record.
4. Request a Trialforce Management Organization (TMO).
5. Optionally, create a customized branded login page and branded emails in your TMO.
6. Create a Trialforce Source Organization (TSO) from your TMO.
7. Install your managed package in the TSO, and customize it as you want your prospects to experience it. You can apply custom branding, load sample data, create custom profiles, and so on.
8. Create a new Trialforce template from the TSO.
9. Link the Trialforce template to the AppExchange.
10. Submit the Trialforce template for security review and get it approved.

You can now use this template to create free trials. For more information, see:

- Providing a free trial on AppExchange
- Providing a free trial on your website
- Providing a free trial using the API

Link a Package with Your License Management Organization

To receive lead and license records from customer installs, link a managed package to your License Management Organization (LMO), the organization where the License Management App (LMA) is installed. You also specify default license settings for your offering during this process. Default license values are used to set the Status, Expiration Date, and Seats fields on the license record in the LMA and in the installer's organization.

Note: When you link a package with an LMO, that package's leads and licenses must be permanently managed out of the LMO. You can't migrate licenses to another organization.

1. Log in to the Partner Community.
2. On the Publishing page, click the **Packages** tab.
3. Find the package that you want to link, and click **Manage Licenses**.
4. Click **Register**.
5. Enter the login credentials for your LMO, and click **Submit**.
6. Select whether your default license is a free trial or active.
7. Enter the license length in number of days. If your license is free or doesn't expire, select **License does not expire**.

8. Enter the number of seats associated with your default license, or select **License is site-wide** to offer the license to all users in the installer's organization.
9. Click **Save**.

To verify that you linked the package successfully, log in to the LMO and click the **Package Versions** tab. After you link a package to your LMO, all versions of that package are associated.

Request a Trialforce Management Org

A Trialforce Management Org (TMO) lets you create and manage Trialforce Source Orgs (TSO) and specify custom branding for your login page and emails. To receive a TMO, you must be a qualified ISV partner, and your offering must have passed the AppExchange security review.

Note: The TMO is separate from your Partner Business Org and the Developer Edition org where you built your offering.

1. Log in to the Partner Community and go to the Support tab.
2. Select **New Case**.
3. Select **Trialforce**, and then select **Create a Case**.
4. For Subtopic, select **Trialforce Management Org**.
5. Enter the required information in the `Description` field, and then select **Submit Case**.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To manage Trialforce:
- Customize Application

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.

Creating Branded Emails

You can customize the branding of the emails sent to subscribers of new trial organizations.

To create a branded email set:

1. Log in to your Trialforce Management Organization.
2. From Setup, enter *Branding* in the Quick Find box, select **Branding**, then click **Email Sets**.
3. Click **New Email Set** or **Edit** next to an existing email set.
4. Enter a name for the email set and your company information.
5. In the Preview Emails area, click through the different types of generated emails and make sure they read correctly.

   Note: The login URL displayed in the preview will always be http://login.salesforce.com even if you use a branded login page. These two processes are distinct.

6. Click **Save**.
7. If you're ready to make these emails available to your Trialforce Source Organization (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

To assign a branded email set to your TSO:

1. From Setup, enter *Source Organizations* in the Quick Find box, then select **Source Organizations**.
2. Click **Edit** next to your TSO.
3. Select the email set.
4. Click **Save**.
5. Click **Login** if you want to see your branded login page in action.

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**.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To manage Trialforce:
- Customize Application

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.

Create a Trialforce Source Organization

A Trialforce Source Organization (TSO) acts as the basis for a new trial org. After you create a TSO, you install your package there. You then add data to give your prospects something to explore when they first log in to the trial org.

You have two options for creating a TSO: You can use a Trialforce Management Organization (TMO) or the Environment Hub. If you plan to brand your emails or login page, use a TMO. When you create the TSO in a TMO, you also get a custom domain (My Domain). Here's how to create a TSO (Enterprise Edition) from a TMO.

Note: If you create a TSO from a TMO, it's always an Enterprise Edition. To create a Professional Edition TSO, create the TSO from the Environment Hub.

EDITIONS

Available in: Salesforce Classic

Available in: **Developer** Edition

USER PERMISSIONS

To manage Trialforce:
- Customize Application

1. Log in to your TMO.
2. From Setup, enter `Source Organizations` in the `Quick Find` box, then select **Source Organizations**.
3. Click **New**.
4. Enter a new username and email address for the administrator account.
5. Enter a name for the TSO. Optionally, specify the custom branding by choosing a branded email set or login site.
6. Click **Create**.

You can also create a TSO from the Environment Hub. When you use the Environment Hub, you can create an Enterprise Edition TSO or a Professional Edition TSO.

1. Log in to the Environment Hub.
2. Click **Create Org**.
3. Keep the default, **Purpose as Trialforce**.
4. Keep the default for Create Using, **Standard Edition**.
5. Select **Professional TSO** or **Enterprise TSO**.
6. Enter the org name.
7. (Optional) Enter a unique name for the My Domain custom domain.
8. Enter a username and email address for the admin account.
9. Enter a name for the TSO.
10. Acknowledge that you've read the Master Subscription Agreement.
11. Click **Create**.

The TSO now appears in the Environment Hub.

You receive an email with the login details for your TSO. You can then log in to the TSO and install your package, along with sample data and configurations. Optionally, you can also create:

- Custom profiles
- New users

- Sample records

The goal is to configure the TSO exactly as you want your customers to experience it. You can then create a Trialforce template, which is a snapshot of your TSO at a specific point in time.

Note: Here are some considerations when working with a TSO.

- Always associate a managed package with the License Management Organization (LMO) before installing the offering in your TSO. If you don't follow that order, trial orgs provisioned from the TSO don't generate leads or licenses in the LMO.
- Before creating a Trialforce template, ensure that the TSO admin has a license for the offering installed in the TSO.
- You can create multiple TSOs from your TMO, so you can set up trials for different products, each with its own configuration and branding.
- All TSOs expire after one year. If you want to use the TSO for a longer period, log a case to request an extension.

SEE ALSO:

How do I configure who can use Lightning Experience in my trial org?

Create a Trialforce Template

Create a Trialforce Template

A Trialforce template is a snapshot of your Trialforce Source Organization (TSO) at a given instance in time. For security reasons, however, Personally Identifiable Information (PII) on the User Object, such as that in the address fields, is scrubbed from templates. PII in custom objects and fields is not modified. Before you create the template, make sure that you've installed your package into the TSO. Then, configure it exactly as you want your customers to experience it, with the appropriate sample data, profiles, users, and records.

EDITIONS

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

Available in: **Developer**, **Professional**, and **Enterprise** Edition

USER PERMISSIONS

To manage Trialforce:
- Customize Application

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

1. Log in to your TSO.
2. From Setup, enter `Trialforce` in the `Quick Find` box, then select **Trialforce**.
3. Click **New Trialforce Template**.
4. Describe the template and any optional features.

   By default, templates are public. To create a private template, select **Mark this template as private so that only authorized orgs can sign up**. You can then indicate which orgs are authorized to sign up new orgs using this template.

   If the template isn't private, the default options are fine for most cases.
5. Click **Save**.
6. (Optional) If you created a private template, enter the org ID of the orgs that can sign up using this template, then click **Save**.

   You can enter up to 51 org IDs, each on a separate line.

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.

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.

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.

Link a Trialforce Template to the AppExchange

To offer a free trial with your app or component listing, link a Trialforce template to the AppExchange.

1. Log in to the Partner Community.
2. On the Publishing page, click the **Organizations** tab.
3. Click **Connect Organization**.
4. Enter the login credentials for the organization that contains the trial template. If you developed multiple trial templates in this organization, they are all linked to the AppExchange.
5. Click **Submit**.
6. Optionally, click the **Trial Templates** tab to view the linked template and create a listing.

Submit a Trialforce Template for Security Review

To offer a trial on the AppExchange using Trialforce, your template must pass a security review. Before requesting a review, link the organization containing your Trialforce template to the AppExchange.

Note: You can only request a review on a Trialforce template that has at least one package installed. You must own or license the installed packages in the template and it should have already passed the security review.

1. Log in to the Partner Community.
2. On the Publishing page, click the **Trial Templates** tab.
3. Next to the template that you want reviewed, click **Start Review**.

You receive an email confirmation after you initiate the review and another email when the review is completed. The review is free for partners and typically takes 2–3 days.

Provide a Free Trial on the AppExchange

To create trials on the AppExchange, your app or component must:

- Be a managed package
- Be managed via the License Management Application
- Autoprovision—that is, the user must not need to interact with you at any point to get the app or component up and running
- Have passed the security review
- Have passed the Trialforce template review

You can provide a free trial on the AppExchange in three ways.

- Using Trialforce
- By configuring a test drive
- By installing your app or component into an existing organization

Provide a Free Trial on the AppExchange Using Trialforce

Providing a free trial lets potential customers experience your offering before purchasing or subscribing.

Note: You must be an eligible partner to provide free trials. For more information on the Partner Program, including eligibility requirements, visit www.salesforce.com/partners.

1. Create a Trialforce template with your offering installed and configured as you want your prospects to experience it. For details, see Setting up Trialforce.
2. Submit the Trialforce template for security review. This review is free and takes less time than the initial review of your app or component.
3. Link the Trialforce template to your AppExchange listing.
   a. Log in to the Partner Community.
   b. On the Publishing page, click the **Listings** tab.
   c. Find the listing where you want to offer a trial, and click it to open the AppExchange publishing console.
   d. Click the **Trials** tab, and select **Offer a free trial organization**.
   e. Follow the on-screen prompts to add a trial template to the listing.
4. Click **Save**.

Now, when customers visit your listing, they can start a free trial with your offering preinstalled, even if they don't have a Salesforce account. If they decide to start a trial, we collect their contact information and ask them to agree to your terms and conditions and our MSA. After they provide this information, prospects receive an email prompting them to log in to a trial organization.

Offer a Test Drive on AppExchange

A test drive lets customers try your product in a Developer Edition org that's preconfigured with sample data. The test drive org has two types of users: an admin and a read-only evaluator. The admin user configures the org for the test drive. The evaluation role lets customers log in to the org and experience your product.

Use the Publishing Console to create test drive orgs. Otherwise, customers can experience issues when logging in as evaluators.

Note: Salesforce doesn't support test drives orgs created outside of the Publishing Console. However, if you create a test drive using another method and your customers experience login issues, try setting profile-level IP login ranges from `0.0.0.0` to `255.255.255.255`. For more information, see "Restrict Login IP Ranges in the Enhanced Profile User Interface" in the Salesforce Help.

1. Log in to the Partner Community.
2. Click **Publishing**.
3. Click **Listings** and then select the product for which you want to offer a test drive.
4. On the Trials tab, select **Offer a Test Drive**.
5. Click **Create Test Drive**.
6. Give the test drive a customer-friendly name, and associate a package.

7. Click **Submit**. Salesforce creates an org and emails you login credentials for the admin and evaluation users.
8. Log in to the test drive org as the admin user and add sample data.
9. Log out of the org and then log in again as the evaluation user to set a password.
10. In the Publishing Console, go to the Trials tab and click **Connect Organization**.
11. Enter the login credentials for the evaluation user and then click **Submit**.
12. Click **Save**.

Provide a Free Trial on the AppExchange When Your Offering Is Installed

You can provide a free trial of your offering by setting the default license settings on your package. When a customer installs the app or component in an existing Salesforce organization, they can use it for the specified trial period.

Provide Free Trials on Your Website

Use HTML forms to drive traffic to your business and show off your solutions to prospective customers. After a prospect submits your form, Salesforce provisions a trial based on your Trialforce template.

To provide a free trial on your website, first set up Trialforce. Then, complete the following tasks and you're ready to go live.

Enable the SignupRequest API
Log a case in the Partner Community to enable the SignupRequest API in your org.

Choose a Sign-Up Form Hosting Option
The sign-up form serves as the registration page that prospective customers use to sign up for trials. Review and choose a hosting option for your sign-up form.

Create Sign-Ups Using the API
Use API calls to the SignupRequest object to create sign-ups for prospective customers.

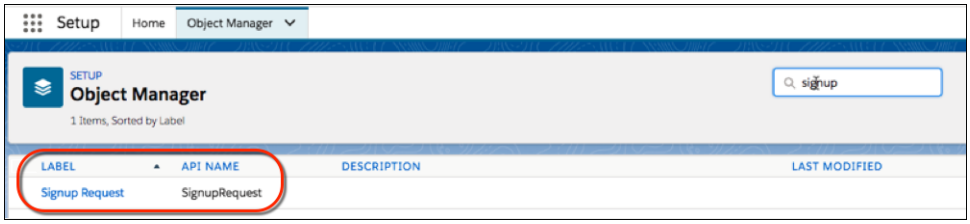
Provision Trial Orgs
Use Trialforce to provision a free trial of your solution for prospective customers.

Enable the SignupRequest API

Log a case in the Partner Community to enable the SignupRequest API in your org.

Tip: Enable the SignupRequest API in your business org. Then, you can easily integrate sign-up data with your existing business processes. For example, create a workflow rule to convert sign-up requests into leads and run reports to track the number of sign-ups in a given period.

1. Check to see if the SignupRequest API is enabled in your business org.
   a. Log in to your business org.
   b. From Setup, enter `Object Manager` in the Quick Find box, then click **Object Manager**.
   c. Verify that the Signup Request object appears. If you don't see this object, log a case to enable the SignupRequest API.



2. Log a case to enable the SignupRequest API.
   a. Log in to the Partner Community.
   b. Under the Support tab, click **New Case**.
   c. In the AppExchange Partner (ISV) Technology category, hover over Trialforce and then click **Create a Case**.
   d. Select a severity level.
   e. Enter a case subject.
   f. In the Subtopic dropdown list, select **Signup API Feature Request**.
   g. Provide the required case details.
   h. Click **Submit Case**.

Choose a Sign-Up Form Hosting Option

The sign-up form serves as the registration page that prospective customers use to sign up for trials. Review and choose a hosting option for your sign-up form.

The SignupRequest API supports several HTML form hosting options. Choose one of the following:

- Node.js and React app hosted on Heroku
- Lightning component hosted on Salesforce Community
- Visualforce page hosted on Sites
- Web-to-Lead form with Process Builder

Make Unauthenticated Calls to the SignupRequest API

By default, the SignupRequest API is available only to authenticated calls. If you have a use case that requires unauthenticated calls, for example, making your sign-up form available to unauthenticated users, follow the pattern in the code sample.

```
public with sharing class newTrialSignupController {
@auraEnabled
public static void getNewLead(Lead newLead, String templateId, String username, Boolean
createLead, String domain) { }
// SignupCreation is an inner class without sharing. It runs in the system context
// and is used to handle SignupRequest calls for unauthenticated users.
public without sharing class SignupCreation {
public void createNewTrial(Lead newLead, String templateId, String username, String domain)
 {
}}}
```

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

SEE ALSO:

Provide Free Trials on Your Website

SignupRequest API

Make it Easy for Your Customers to Provision Trials Part 1

Make it Easy for Your Customers to Provision Trials Part 2

Web Form Replacement Code in Nodejs and React

Demo App to Create Trial Orgs Using the SignupRequest API

Provision Trial Orgs

Use Trialforce to provision a free trial of your solution for prospective customers.

Once you've configured Trialforce, you can provision trial orgs two ways.

- Push—You provision a trial on behalf of a prospective customer by filling out the registration form with your prospect's information.
- Pull—A prospect requests a trial on their own by filling out a registration form on your public website.

1. Upload the HTML registration form to your public web servers.
2. Edit and publish the appropriate HTML pages on your company website where you want to include a link to the Trialforce registration form.
3. Navigate to the registration page from your company website.
4. Fill in the required fields and submit the form.

Anyone with access to the form can create a trial on behalf of a prospect without the need to expose the form on the company website. Just launch the registration form HTML file in a browser, fill in the fields on behalf of the customer, and submit the form. Your prospect receives an email, optionally branded with your company information, indicating the new trial is available.

Modify the Trial for an Upgrade

You can update your trials to reflect changes to your offering or its custom branding. To do so, you must:

- Create and publish a new version of your managed package (or an extension package).

- Have a Trialforce Source Organization (TSO) where you can upload the new package version. You can reuse the TSO that you used to create your original Trialforce template, or you can create a new one. If you create a new TSO, be sure to link it to the AppExchange.

1. Install your updated managed package (or extension package) into your TSO.
2. Make any other desired changes in the TSO, such as loading sample data or updating custom branding.
3. Create a Trialforce template for your trial.
4. Submit the template for review.
5. For trials created using an HTML sign-up form or using the API, complete the following steps.

Trial Method	Steps
Via an HTML form on your company website	Log a case to associate the new Trialforce template with the sign-up form. **a.** Log in to the Partner Community. **b.** Under the Support tab, click **New Case**. **c.** Select the **AppExchange and Feature Requests** > **Trialforce** category. **d.** For the description, provide the TSO ID, the ID of the new Trialforce template, and the name of your sign-up form.
Using the API	Log a case to get the template approved for SignupRequest API use. **a.** Log in to the Partner Community. **b.** Under the Support tab, click **New Case**. **c.** Select the **AppExchange and Feature Requests** > **Trialforce** category. **d.** For the description, provide the TSO ID, the ID of the new Trialforce template, and the organization to use for creating sign-ups.

Trialforce Best Practices

Here are some guidelines for using Trialforce.

- Create several Trialforce Source Organizations (TSOs) for customized trial experiences, for example, one for each managed package, industry vertical solution, country.
- Load sample data into the TSO.
- Apply custom branding to your trial signup form, log-in page, and emails.
- Update your Trialforce template each time you release a new version of your app.
- Once you've set up Trialforce, go through the signup flow to confirm that everything is working as you expect it to. This testing can help you identify areas to where you can improve the signup process.

Although Trialforce was primarily designed for enabling free trials, it's also useful in other contexts. For example, you can use it to:

- Create trial organizations for sales demos.
- Create test organizations with sample data for internal QA.

Trialforce FAQ

This section contains a list of frequently asked questions about Trialforce.

- How do I upgrade my trial with a new version of my offering?
- Can I distribute my app or component using both Trialforce and the AppExchange?
- How are trials different from Trialforce?
- Is it possible to install another app in a trial organization?
- How do I configure who can use Lightning Experience in my trial org?

How do I upgrade my trial with a new version of my offering?

Install the new version of the package into your Trialforce source organization. After upgrading, create a new Trialforce template and use the template as the basis for your trial.

Can I distribute my app or component using both Trialforce and the AppExchange?

Of course! The most effective way to distribute your offering is by using Trialforce and the AppExchange together. You can even advertise your Trialforce page on your AppExchange listing and vice versa. Generally, the AppExchange is best for engaging existing Salesforce customers, while Trialforce works great with new customers.

How are trials different from Trialforce?

Trials are administered from the AppExchange whereas Trialforce is administered from your own website.

Is it possible to install another app in a trial organization?

Yes. The Trialforce master organization is a fully functioning Salesforce organization. Your customer will have your app installed and can subsequently install additional apps into the same organization as they see fit. It's just like any other free trial of Salesforce.

How do I configure who can use Lightning Experience in my trial org?

The future of the Salesforce user experience and Salesforce platform is Lightning Experience. With Lightning Experience, you can move faster, do more, and be more productive. Going forward, all our new features and innovations will be in Lightning Experience.

You can remove the ability for users with a specific profile to switch back to Salesforce Classic.

1. From Setup, enter `Profiles` in the Quick Find box, then select **Profiles**.
2. Select the profile you want to keep using Lightning Experience.
3. On the profile detail page, click **Edit**.
4. Deselect the **Hide Option to Switch to Salesforce Classic** permission.

By default, new orgs are configured to switch Salesforce Classic users to Lightning Experience once a week. Only users with the Lightning Experience User permission are affected. You can configure or disable automatically switching users to Lightning Experience.

1. From Setup, enter `Lightning` in the Quick Find box, then select **Lightning Experience**.

2. On the Set Up Users tab in the Migration Assistant, turn off **Encourage Users to Stay in Lightning Experience**, or specify how often users are switched to Lightning Experience.

You can disable Lightning Experience for some profiles.

1. From Setup, enter `Profiles` in the Quick Find box, then select **Profiles**.
2. Select the profile you want to change.
3. On the profile detail page, click **Edit**.
4. Deselect the **Lightning Experience User** permission.

You can disable Lightning Experience in a new Trialforce Source org so that new users default to Salesforce Classic.

1. From Setup, enter `Lightning` in the Quick Find box, then select **Lightning Experience**.
2. On the Turn It On tab in the Migration Assistant, click the button to switch it to **Disabled**.

CHAPTER 12 Support Your AppExchange Customers

In this chapter ...

- Subscriber Support Console
- Usage Metrics

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 Support Console

Using the Subscriber Support Console, you can easily access information about all your subscribers, such as which Salesforce Edition they are using and if they are over their limits. Subscribers can also grant you login access to troubleshoot issues directly within the app, in the familiar manner that they grant login access to administrators. Once granted access, you can log in to the subscriber's organization and directly view their configuration and data to help troubleshoot problems.

Note: This feature is available to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, please visit us at www.salesforce.com/partners.

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.

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.

Logging 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 only available in orgs with a Salesforce Platform or full Salesforce license. You can't log in to subscriber orgs on Government Cloud instances.

To log in, once a user has granted you access:

1. In the License Management App (LMA), click the **Subscribers** tab.
2. To find a subscriber org quickly, 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, from Setup, click **Return to Subscriber Overview** to return to your org.

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

Best Practices

- 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, 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 Customer Debugger

Each License Management Org can use one free ISV Customer Debugger session at a time. The ISV Customer Debugger is part of the Apex Debugger, which is part of the Lightning Platform IDE plug-in for Eclipse and is normally a paid feature. The Apex Debugger can be used only in sandbox orgs, so you can initiate debugging sessions only from a customer's sandbox.

For information on the ISV Customer Debugger, see *Force.com IDE Developer Guide*: Get Started with the ISV Customer Debugger.

SEE ALSO:

Salesforce Help: Open the Developer Console

Usage Metrics

You can collect detailed usage metrics from each organization in which your managed package is installed. By analyzing this information, you can gain valuable insights into the utilization and performance of your app across your entire customer base. For example, you can identify:

- The features most and least used — this can help you prioritize your development efforts when planning the next version of your app.
- The customers using your app most intensively — these are your most valuable customers.
- The customers whose usage of your app is minimal or declining — these are the customers most at risk of attrition.

EDITIONS

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

You can collect the following daily metrics on two types of components in a managed package.

- **Custom objects** — the total number of records existing per organization in each custom object. This enables you to track how the usage of that custom object is growing with time in any subscriber organization, which is a reliable indicator of how much it's being utilized
- **Visualforce pages** — the number of times per organization each Visualforce page was accessed, the number of unique users who accessed it, and the average loading time (in milliseconds). By comparing the metrics for different Visualforce pages, you can determine the relative popularity of different parts of your app in a specific customer organization, as well as trends across all customers.

The custom objects data is a snapshot that reflects the state of the organization at the time the database was sampled, while the Visualforce data covers usage over a 24-hour period.

The usage metrics data for all production organizations in a given instance is merged and written into a text file, in a specified format, once a day. Currently, no data is collected on packages installed in sandbox organizations or on managed beta packages.

This feature is intended for API access only. You must write a custom process to collect the metrics data from the reporting organization, and export it to a system of your choice for analysis. This gives you the maximum flexibility to monitor and analyze the usage trends most relevant for your app.

Your customers' consent is not required for usage data to be collected, and there's no way for them to opt out. This ensures you receive complete data for your entire customer base. Allowing some users to be excluded would skew the results, making the data less useful.

Note: If any of your customers have concerns about privacy, reassure them any data collected is limited to usage statistics. No customer data is ever exposed to the ISV under any circumstances. This is consistent with salesforce.com's emphasis on trust as a core value.

Setting up Usage Metrics

To set up Usage Metrics for any package, two organizations have special importance.

- **Release organization** — the Development Edition organization used to upload the package.
- **Reporting organization** — the organization to which the usage data is delivered, on a daily basis.

The release organization and reporting organization must be members of the same Environment Hub. This is a security feature, to ensure usage data is only delivered to an organization controlled by the developer of the package. We recommend using the Environment Hub as your reporting organization.

To set up Usage Metrics for a package:

1. Set up Environment Hub, if you haven't already done so.
2. Connect the release organization to the Environment Hub.
3. Connect the reporting organization to the Environment Hub (if they're different).
4. Log a case in the Partner Community to activate Usage Metrics. You'll need to provide the package ID for your app.

Once the feature is activated, you'll receive a confirmation email. From that point on, usage data will automatically be collected from all organizations in which your package is installed, and delivered to the reporting organization on a daily basis. There is no way to get usage data retroactively, that is, for any period prior to the activation of Usage Metrics.

Accessing Usage Metrics Data

The usage data for a package is stored in `MetricsDataFile` records in your reporting organization. Once you activate the Usage Metrics feature, one new record is created for all custom objects and one for all Visualforce pages, per Salesforce instance per day.

Note: To see the number of Salesforce instances currently in use, visit trust.salesforce.com.

The usage data for each day and instance is stored as a text file, encoded in Base 64, in the `MetricsDataFile` field of the record. Other fields in the record identify these properties.

- Namespace prefix of the package
- Salesforce instance
- Start time and date of data collection
- End time and date of data collection
- Size of the data file in bytes
- Type of data, which is either `CustomObject` or `Visualforce`

The custom objects data is a snapshot that reflects the state of the organization at the time the database was sampled, while the Visualforce data covers usage over a 24-hour period.

The custom object count is a snapshot captured once each day. Here's a section of a sample data file for custom objects. It shows there were 3500 and 1500 records in the `Alpha` and `Beta` custom objects, respectively, in the specified customer organization on the specified day.

```
"00Dxx0000001gbk","org1","Enterprise Edition","TRIAL","Alpha", "3500"
"00Dxx0000001gbk","org1","Enterprise Edition","TRIAL","Beta", "1500"
```

In a record for Visualforce pages, each row of the text file contains usage data in the following order.

- Organization ID
- Organization name
- Organization edition
- Organization status
- Package version number

- Name of the Visualforce page
- Number of times the page was accessed
- Number of unique users who accessed the page
- Average loading time of the page, in milliseconds

The Visualforce counts for each organization measure the number of times the page was viewed in the duration between the start and end times. Here's a section of a sample data file for Visualforce pages.

```
"00Dxx0000001gbk","org1","Enterprise Edition","TRIAL","1.0","/apex/gm12__f1","1","1","66.0"
"00Dxx0000001gbk","org1","Enterprise Edition","TRIAL","1.0","/apex/gm12__f2","1","1","128.0"
"00Dxx0000001gbk","org1","Enterprise Edition","TRIAL","1.0","/apex/gm12__f3","1","1","107.0"
"00Dxx0000001gbf","org1","Enterprise Edition","TRIAL","1.0","/apex/gm12__f1","5","1","73.6"
"00Dxx0000001gbf","org1","Enterprise Edition","TRIAL","1.0","/apex/gm12__f2","1","1","72.0"
"00Dxx0000001gbf","org1","Enterprise Edition","TRIAL","1.0","/apex/gm12__f3","7","1","50.8"
```

You must write a custom process to query the reporting organization to collect the metrics data, and export it to a system of your choice for analysis. This gives you the maximum flexibility to monitor and analyze the usage trends most relevant for your app.

MetricsDataFile

Represents a data file containing usage metrics on all installations of a managed package in a Salesforce instance. This object is available in API version 30.0 and later.

Supported Calls

`query()`, `delete()`

Fields

Field Name	Details
`MetricsDataFile`	**Type** base64 **Properties** Filter, Query, Sort **Description** A text file containing the usage data encoded in Base 64.
`MetricsDataFileContentType`	**Type** string **Properties** Filter, Query, Sort **Description** The format of the data file. Currently, the only allowed value is `text/csv`.
`MetricsDataFileLength`	**Type** int

Field Name	Details
	Properties Filter, Query, Sort **Description** The size of the data file in bytes.
`MetricsRunDate`	**Type** dateTime **Properties** Filter, Query, Sort **Description** The date when the usage metrics collection job was run.
`MetricsEndDate`	**Type** dateTime **Properties** Filter, Query, Sort **Description** The end time and date for the data collection.
`MetricsStartDate`	**Type** dateTime **Properties** Filter, Query, Sort **Description** The start time and date for the data collection.
`MetricsType`	**Type** picklist **Properties** Filter, Query, Sort **Description** The type of data being collected. The possible values are `CustomObject` and `Visualforce`.
`NamespacePrefix`	**Type** string **Properties** Filter, Query, Sort **Description** The namespace prefix of the package for which data is being collected.

Field Name	Details
SendingInstance	**Type** string **Properties** Filter, Query, Sort **Description** The server instance from which this data was collected, for example, "na8."

Usage

Use this object to access customer usage metrics for a managed package. Each record contains one day's data, on either custom objects or Visualforce pages, for all organizations in a Salesforce instance that have the package installed. The following data is collected each day.

- **Custom objects** — the number of records stored in each custom object.
- **Visualforce pages** — the number of times each Visualforce page was accessed, the number of unique users who accessed it, and the average loading time (in milliseconds).

Usage Metrics Visualization

The Usage Metrics Visualization app, available from Salesforce Labs on the AppExchange, enables you to visualize trends in usage metrics data for your app. You can use the Usage Metrics Visualization app to generate charts showing changes in various app metrics, over a specified duration, for one or more customer organizations.

The app must be installed in your Usage Metrics reporting organization and requires Usage Metrics to be enabled in advance, so some data is available for analysis. You can analyze data going back a maximum of 30 days. If Usage Metrics wasn't enabled for the entire time period that you specify, only partial data is plotted.

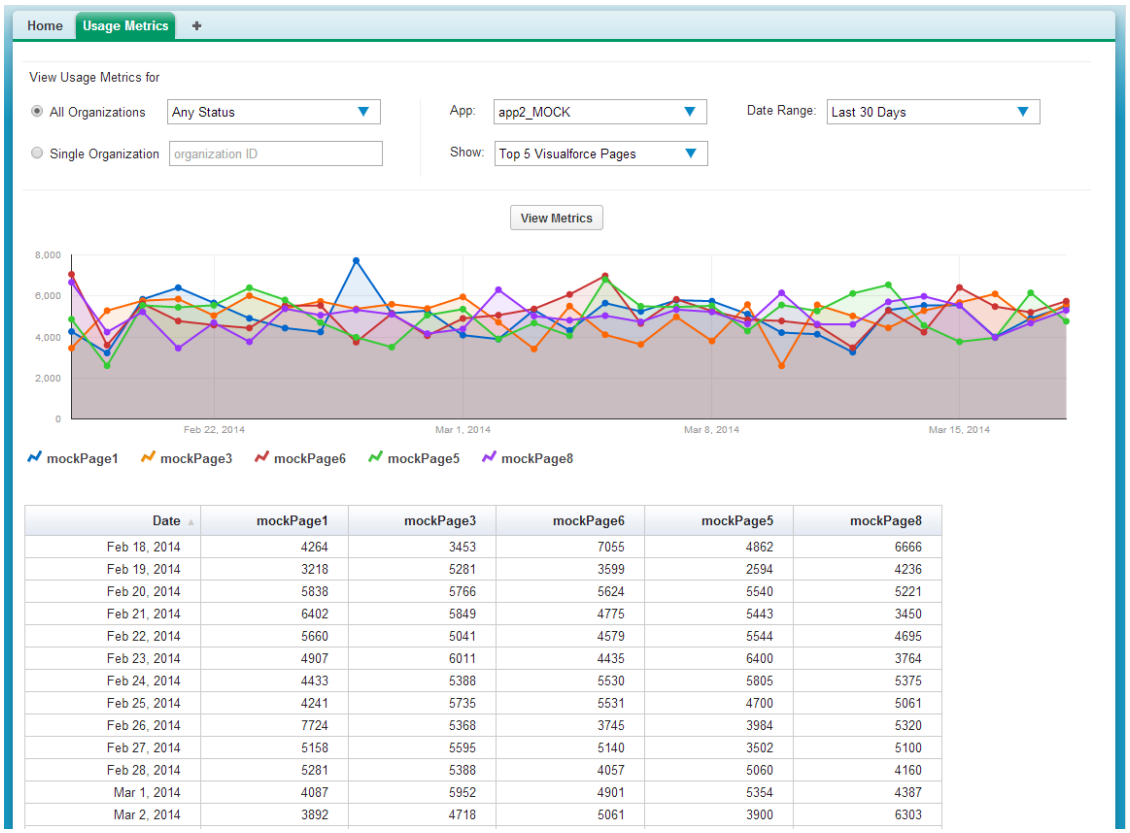
The app is intended as a reference implementation, for illustration purposes only. It's distributed as an unmanaged package, so you can review its components and extend or customize it to meet your requirements. If your visualization needs are more complex, you can export the raw metrics data from the reporting organization and analyze it by using custom code or a third-party tool .

To install the Usage Metrics Visualization app:

1. Go to the AppExchange and search for the Usage Metrics Visualization app.
2. Click **Get It Now**.
3. Enter the credentials for your reporting organization, and then click the login button.
4. Click **Install**.

You'll see a message describing the progress and a confirmation message after the installation is complete.

The Usage Metrics Visualization app showing data for the top five Visualforce pages.



Date	mockPage1	mockPage3	mockPage6	mockPage5	mockPage8
Feb 18, 2014	4264	3453	7055	4862	6666
Feb 19, 2014	3218	5281	3599	2594	4236
Feb 20, 2014	5838	5766	5624	5540	5221
Feb 21, 2014	6402	5849	4775	5443	3450
Feb 22, 2014	5660	5041	4579	5544	4695
Feb 23, 2014	4907	6011	4435	6400	3764
Feb 24, 2014	4433	5388	5530	5805	5375
Feb 25, 2014	4241	5735	5531	4700	5061
Feb 26, 2014	7724	5368	3745	3984	5320
Feb 27, 2014	5158	5595	5140	3502	5100
Feb 28, 2014	5281	5388	4057	5060	4160
Mar 1, 2014	4087	5952	4901	5354	4387
Mar 2, 2014	3892	4718	5061	3900	6303

To visualize the usage metrics data:

1. Specify the app whose metrics you want to view by selecting it from the App menu.

   Note: You should have enabled Usage Metrics for your app at least a few days before, so some usage data is available to analyze.

2. Specify the organization(s) that you want to view metrics for by choosing one of these options.
   - For a single organization, enter its Organization ID in the Single Organization field.
   - For a group of organizations, select one of the following from the All Organizations menu.
     - Any Status
     - All Active: These are organizations used by paying customers.
     - All Free: These are Developer Edition (DE) organizations.
     - All Trial: These are trial organizations, which expire after a specified period.

3. Specify the type of metric that you want to visualize by selecting one of these values from the Show menu.
   - Total Visualforce Page Views
   - Top 5 Visualforce Pages
   - Total Record Count
   - Top 5 Objects by Record Count

4. Specify the time period to cover by selecting one of these values from the Date Range menu.

- Last 30 Days
- Last 7 Days
- Last 2 Days

Note: If the volume of usage data is too large, you might get an error message. In that case, choose a smaller date range and try again.

5. Click **View Metrics**.

The data you specified is displayed on the page as a chart and as a table. To visualize a different data set, change the parameters, and then click **View Metrics** again.

CHAPTER 13 Update Your Solution

In this chapter ...

- About Package Versions
- Create and Upload Patches
- Working with Patch Versions
- Publish Upgrades to Managed Packages
- Pushing an Upgrade

Your packaged solution is ready for an update. Learn how to fix small issues with patches and make major changes with upgrades.

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.

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

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.

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.

Create and Upload Patches

Note: Patch versions and push upgrades are only available to Salesforce ISV partners.

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**.

   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**.

To distribute your patch, you can either share the upload link or schedule a push upgrade.

Working with Patch Versions

Note: Patch versions and push upgrades are only available to Salesforce ISV partners.

A *patch version* enables a developer to change the functionality of existing components in a managed package. Subscribers experience no visible changes to the package. Patches are minor upgrades to a Managed - Released package and only used for fixing bugs or other errors.

Patch versions can only be created for Major Releases. Subscribers can receive patch upgrades just like any other package version. However, you can also distribute a patch by using push upgrades.

When you create a patch, the `patchNumber` on a package's `Version Number` increments by one. For example, suppose that you release a package with the version number 2.0. When you release a patch, the number changes to 2.0.1. This value can't be changed manually.

Patch Development Organizations

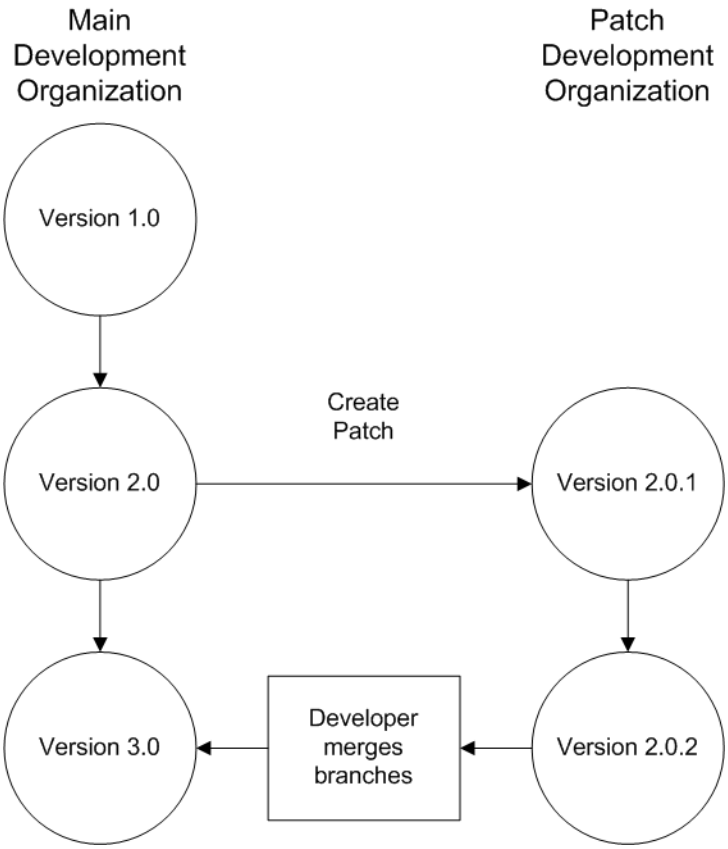
Every patch is developed in a *patch development organization*, which is the organization where patch versions are developed, maintained, and uploaded. To start developing a patch, create a patch development organization. See Create and Upload Patches. Patch development organizations are necessary to permit developers to change existing components without causing incompatibilities between existing subscriber installations.

A patch development organization can upload an unlimited number of patches. Only one patch development organization can exist per major release of your package. A patch development organization for a package with a version number of 4.2 can only work on patches such as 4.2.1, 4.2.2, 4.2.3, and so on. It won't work on version 4.1 or 4.3.

Integrating Patch Development

The following diagram illustrates the workflow of creating a patch and integrating any work into future versions:

Patch Development Workflow



After version 2.0 is released, the developer creates a patch. The package version number in the patch development organization starts at 2.0.1. As the main development organization moves towards a released version of 3.0, a second patch is created for 2.0.2. Finally, the developer merges the changes between the main development organization, and the patch development organization, and releases the package as version 3.0.

If you are developing your packages using the Force.com IDE, you can take advantage of revision control systems in Eclipse to compare and merge different project branches.

Salesforce recommends using the Subversion plug-in. To install Subversion for the IDE:

1. Go to `http://subclipse.tigris.org` to get the latest Eclipse Update Site URL compatible with your version of Eclipse.
2. In the Lightning Platform IDE, navigate to **Help** > **Software Updates**, and select the Available Software tab. Click **Add Site**, and enter the URL from the previous step.
3. Select the new site and click **Finish** to fetch the latest version of the Subclipse plug-in. Select the required Subclipse plug-in from the list returned from the site.
4. Click **Next**, accept the terms, and click **Next** again.
5. To begin the installation, click **Finish** and then **Install All**. Restart Eclipse after the installation completes.

You have now linked your Lightning Platform IDE environment to Subclipse. The next step is to connect your repository to the environment:

1. Open the `SVN Repository Exploring` perspective in the IDE. Locate the SVN Repositories view.
2. Use the **Add SVN Repository** icon on the far right to configure Subclipse to access the local repository. The URL to access your repository locally is `file:///svn_repos`.

The subversion repository tracks changes made to stored projects. Because working with patches involves two different branches, a main development organization and a patch development organization, you combine your changes for a future release. To view the different versions of your package:

1. Open the `Project Explorer` perspective.
2. Navigate to a file in your main development project that you want to compare, and use the context menu to select **Compare With...** > **Branch/Tag...**.
3. In the `Compare to` field, select the patch version of the file.
4. Click `Graphical`, then click **OK**.

The changes between the main development organization's file and the file stored in the patch development organization are highlighted. You can use this view to merge any differences between the two projects.

For more information on using the Lightning Platform IDE, see the *Platform Developer's Guide*.

Versioning Apex Code

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 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 calling the `System.requestVersion` method. In this way, package developers can determine the request context and specify 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){

        // Add a new validation to the package
        // Applies to versions of the managed package greater than 1.0
        if (System.requestVersion().compareTo(new Version(1,0)) > 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');
        }
    }
}
```

To compare different versions of your Apex classes, click the **Class Definition** tab when viewing the class details.

For more information about the `System.requestVersion` method, see the *Apex Developer Guide*.

Apex Deprecation Effects for Subscribers

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`.	
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.	

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 administrator should manually apply any acceptable changes after installing an upgrade. For more information, see About API and Dynamic Apex Access in Packages on page 63.

To publish upgrades to a managed package:

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.

   Note: 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. For more information, see Manage Versions on page 285.

   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.

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

Delete Components in Managed Packages

After you've uploaded a Managed - Released package, you may find that a component needs to be deleted from your org. The options available for deleting a component depends on the situation in which you are trying to delete the component.

One of the following situations may occur:

- The component, after 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, and can be undeleted from either the Deleted Package Components page or through the Recycle Bin.

USER PERMISSIONS

To delete components from a package:
- Create AppExchange Packages

Note:

- To enable component deletion in your packaging org, log a case in the Partner Community .
- Because the managed package components behavior differs from the behavior of public Apex classes and public Visualforce components, delete Visualforce pages and global Visualforce components from a managed package using a two-stage process. When you upgrade a package in a subscriber org, the Visualforce pages and global Visualforce components that you deleted aren't removed. Although a Delete button or link is available to org administrators, many orgs continue using obsolete pages and components. However, public Apex classes and public Visualforce components are deleted as part of the upgrade process. If you delete pages and components without performing this two-stage procedure, Salesforce can't warn you when later deletions of public classes and components break your subscribers' obsolete pages and components.

  If you're deleting a Visualforce page or global Visualforce component that refers to or uses public Apex classes or public Visualforce components, perform the deletion steps in this order.

  1. Stage one: Remove references.
     i. To remove all references to public Apex classes or public Visualforce components, edit your Visualforce page or global Visualforce component .
     ii. Upload your new package version.
     iii. Push the stage-one upgrade to your subscribers.
  2. Stage two: Delete your obsolete pages or components.
     i. Delete your Visualforce page or global Visualforce component.
     ii. Optionally, delete other related components and classes.
     iii. Upload your new package version.
     iv. Push the stage-two upgrade to your subscribers.

Here are some key types of components you can delete when updating a previously released managed package.

- Custom buttons or links
- Custom console
- Custom fields
- Custom objects
- Custom settings
- Custom tabs
- Field sets
- Permission sets
- Record types
- S-Controls
- Static resources
- Validation rules
- Visualforce components
- Visualforce pages

For a complete list, see Available Components on page 29.

When you delete a component, you also permanently delete the data that exists in that component. Delete tracked history data is also deleted, and integrations that rely on the component, such as assignment or escalation rules, are changed. After you delete a component in a managed package, you can't restore it or create another component with the same name.

Note: In a managed package, the API names of fields must be unique and cannot be reused even after you delete the component. This restriction prevents conflicts during package installation and upgrade.

Data and metadata are never deleted in a subscriber org without specific action by the customer. When a subscriber upgrades to the new package version, the deleted components are still available in the subscriber's org. The components are displayed in the Unused Components section of the Package Details page. This section ensures that subscribers have the opportunity to export data and modify custom integrations involving those components before explicitly deleting them. For example, before deleting custom objects or fields, customers can preserve a record of their data from Setup by entering `Data Export` in the Quick Find box and then selecting **Data Export**.

Note: It's your responsibility to educate your customers about the potential impact from any components you delete. List all custom components you've deleted and notify customers of any necessary actions, in the Release Notes for your upgraded package.

The following restrictions apply when deleting managed components.

- A component of any type is not deletable if it's referenced by any other metadata, such as workflow rules, validation rules, or Apex classes.
- A custom object is not deletable if it includes any of the following: Apex Sharing Reason, Apex Sharing Recalculation, Related Lookup Filter, Compact Layout, or Action.
- Salesforce doesn't recommend deleting a custom field that is referenced by a custom report type in the same package. Such a deletion leads to an error when installing the upgraded package.
- When you delete a field that is used for bucketing or grouping in a custom report type that's part of a managed package, you receive an error message.
- When you remove a connected app that is a component of a package, the app remains available until you update the package with a new version. 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 cannot 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.

You can delete managed components either declaratively from the user interface or programmatically using Metadata API. With Metadata API, specify the components you want to delete in a `destructiveChanges.xml` manifest file and then use the standard `deploy()` call. The process is identical to deleting components that aren't managed. For more information, see the *Metadata API Developer Guide*.

Viewing Deleted Components

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.

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

Warning: Although a component is deleted, its **Name** remains within Salesforce. You can never create another component with the same name. The Deleted Package Components page lists which names 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**. If a component can be retrieved through the Recycle Bin, it can also be retrieved through this page. You can retrieve the following types of components from here.

- Apex classes and triggers that don't have `global` access.

- Custom tabs.
- 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 permissions
  - 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 the Flows action in Process Builder instead.
- 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.

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

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

Pushing an Upgrade

A *push upgrade* is a method of automatically upgrading your customers to a newer version of your package. This feature works with managed packages only and can be used to ensure that all your customers are on the same or latest version of your package. You can push an upgrade to any number of organizations that have installed your managed package.

A package subscriber doesn't need to do anything to receive the push upgrade. The only indication a subscriber receives after a successful push upgrade is that the package's `Version Number` on the Package Detail page has a higher value. The developer initiating the push resolves upgrades that fail. You can also exclude specific subscriber orgs from the push upgrade by entering the org IDs, separated by a comma, in the Push Upgrade Exclusion List.

Push upgrades minimize the potential risks and support costs of having multiple subscribers running different versions of your app. You can also automate many post-upgrade configuration steps, further simplifying the upgrade process for your customers.

Note: This feature is available to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, visit us at www.salesforce.com/partners.

Push Upgrades

Note: Registered ISV partners can request Push Major Upgrade functionality by logging a case in the Partner Community.

You can push either a patch or a major upgrade. A patch only contains bug fixes and minor enhancements. In contrast, a major upgrade can include major enhancements and new features that add new components. At a high level, pushing an upgrade involves the following steps:

- Upgrade your managed package installed in a customer organization from version X to version Y
- Select one, many, or all customer organizations to upgrade and select a particular version to upgrade to
- Schedule the upgrade to start at a particular date and time
- View progress of upgrades, abort upgrades in progress, or view the result of a push upgrade
- In conjunction with push, you can use a post-install Apex script to automate post-upgrade configurations that your customers have previously performed manually

Warning: When you push an upgrade, you're making changes to a subscriber's org without explicit consent. Therefore, it's important to plan ahead and exercise caution. You can also exclude specific subscriber orgs from a push upgrade by entering the org IDs, separated by a comma, in the Push Upgrade Exclusion List.

Pushing a major upgrade entails a higher degree of risk as it can impact existing functionality in a subscriber's organization. This is because new components in the upgraded package might not be available to existing users of the package, or could overwrite users' customizations. As the app developer, it's your responsibility to protect users from any adverse impact due to upgrading. We strongly recommend you consider all potential consequences of the upgrade and take appropriate steps to prevent any problems.

When pushing a major upgrade, we recommend that you divide changes in your package into two categories:

1. Enhancements to existing features that users already have access to—Use a post install Apex script to automatically assign the relevant components to existing users. This ensures all current users of the package can continue using it without explicit action by administrators.
2. New features you're introducing for the first time—Don't use a post install Apex script to auto-assign components. This ensures your subscribers have the opportunity to decide if and when to use the new features.

Here are some additional guidelines to keep in mind when planning a push upgrade.

- Avoid changes to validation rules, formula fields, and errors thrown from Apex triggers, as they may negatively impact subscribers' integrations.
- Don't make visible changes to a package in a patch. This is because other than a change in the package version number, subscribers aren't notified of push upgrades.
- Test your upgraded package in multiple environments, replicating all relevant features of your customers' organizations, including editions, customizations, other installed packages, and permission sets.
- Schedule push upgrades at your customers' off-peak hours and outside of Salesforce's major release windows, to minimize potential subscriber impact.
- Notify your subscribers in advance about the timing of the upgrade, its potential consequences, and any steps they need to take.

Push Upgrade Best Practices

Push Upgrade is one of the most powerful features we provide to our partners. You have the power to upgrade your customers, but it's imperative that you use that power carefully. Pushing an upgrade without proper planning and preparation can result in significant customer satisfaction issues. Hence, we strongly recommend that you adhere to the best practices documented here.

Plan, Test, and Communicate

- Communicate, communicate, and communicate! Your customers might not even know about the Push Upgrade feature. Some might have strong reservations about changes being pushed to their organizations. Reach out to them and explain how the cloud-computing model works, how they can benefit from seamless upgrades, how you are using best practices to ensure a smooth upgrade, and what your process and commitment is to them regarding the timing and content of an upgrade. Timely and thorough communication is critical for the success of this program.
- Share an upgrade timeline plan with your customers so they know when you will upgrade, and how often.
- Plan when you want to push upgrades to your customers' organizations. Keep in mind that most customers don't want changes around their month-end, quarter-end, and year-end or audit cycles. Do your customers have other critical time periods when they don't want any changes to their organization? For example, there might be certain times when they don't have staff available to verify changes or perform any required post-installation steps.
- Schedule push upgrades during your customers' off-peak hours, such as late evening and night. Have you considered time zone issues? Do you have customers outside the United States who have different off-peak hours? You can schedule push upgrades to any number of customer organizations at a time. Consider grouping organizations by time zone, if business hours vary widely across your customer base.
- Don't schedule push upgrades close to Salesforce-planned maintenance windows. In most cases, it might be better to wait 3-4 weeks after a major Salesforce release before you push major upgrades.
- Test, test, and test! Since you're pushing changes to the organization instead of the customer pulling in changes, there is a higher bar to ensure the new version of your app works well in all customer configurations.

Stagger the Push

- Don't push changes to all customers at once. It's important to ensure that you have sufficient resources to handle support cases if there are issues. Also, it's important that you discover possible issues before your entire customer base is affected.
- Push to your own test organizations first to confirm that the push happens seamlessly. Log in to your test organization after the push upgrade and test to see that everything works as expected.
- When applicable, push to the sandbox organizations of your customers first before pushing to their production organizations. Give them a week or more to test, validate, and fix in the sandbox environment before you push to their production organizations.
- Push upgrades to small batches of customer production organizations initially. For example, if you have 1,000 customers, push upgrades to 50 or 100 customers at a time, at least the first few times. Once you have confidence in the results, you can upgrade customers in larger batches.

Focus on Customer Trust

- You're responsible for ensuring that your customers' organizations are not adversely affected by your upgrade. Avoid making changes to the package, such as changes to validation rules or formula fields, that might break external integrations made by the customer. If for some reason you do, test and communicate well in advance. Please keep in mind that you can impact customer data, not just metadata, by pushing an upgrade that has bugs.
- Write an Apex test on install to do basic sanity testing to confirm that the upgraded app works as expected.
- If you're enhancing an existing feature, use a post-install script to automatically assign new components to existing users using permission sets.
- If you're adding a new feature, don't auto-assign the feature to existing users. Communicate and work with the administrators of the customer organization so they can determine who should have access to the new feature, and the timing of the roll-out.

Assigning Access to New Components and Fields

If the new version of your package includes new components or new fields in existing components, existing users of the package won't automatically have access to the new components and fields after the upgrade. This can limit them from using the new features you've added or prevent older features from working properly. By default, any new components in your package are assigned only to administrators. You have two options for ensuring that all users of the package have access to the new components and fields.

Notify administrators to assign the appropriate permissions to all users of the package
We recommend this for any new features you're introducing. This ensures administrators have the option of deciding if and when to make the new features available.

Assign the new components to existing users automatically, using a post install Apex script
We recommend this for enhancements to existing features. This ensures all current users of the package can continue using it without explicit action by administrators.

To assign access to new components automatically, you can use the following strategy.

1. Create new permission sets that define the default access settings for all new components and fields.
2. Include the new permission sets in the new package version.
3. Write a post install Apex script to run automatically in the subscriber organization after the package is upgraded. The script must perform these tasks.
   a. For each new permission set, choose an existing component whose user assignment needs to be copied.
   b. Find all profiles that can access that component.
   c. Assign the new permission sets to every user with those profiles.

Note: The default permission sets for all standard profiles aren't editable. Hence, the post install script will trigger an exception if it tries to update one of these permission sets. It's important that you create a new permission set to assign access to the new components in your package.

Sample Post Install Script for a Push Upgrade

This section shows a sample post install script that automates the assignment of new components to existing users of a package. For more information on writing a post install Apex script, see Running Apex on Package Install/Upgrade on page 130.

The sample script covers a scenario in which an ISV is upgrading subscribers to a new package version that contains new Visualforce pages and a new permission set that grants access to those pages. After upgrading, existing users of the package will not have access to the new pages by default. The post install script resolves this problem by identifying which users have access to the Visualforce pages in the old version of the package and granting them access to the new pages. The script performs the following actions.

- Get the Id of the Visualforce pages in the old version of the package
- Get the permission sets that have access to those pages
- Get the list of profiles associated with those permission sets
- Get the list of users who have those profiles assigned
- Assign the permission set in the new package to those users

```
global class PostInstallClass implements InstallHandler  {
  global void onInstall(InstallContext context) {

    //Get the Id of the Visualforce pages
    List<ApexPage> pagesList =[SELECT Id FROM ApexPage WHERE NamespacePrefix =
    'TestPackage' AND Name = 'vfpage1'];
```

```
    //Get the permission sets that have access to those pages
    List<SetupEntityAccess> setupEntityAccessList = [SELECT Id,
      ParentId, SetupEntityId, SetupEntityType FROM SetupEntityAccess
      where SetupEntityId IN:pagesList];
    Set<ID > PermissionSetList = new Set<ID> ();

    for(SetupEntityAccess sea : setupEntityAccessList){
    PermissionSetList.add(sea.ParentId)};
    List<PermissionSet> PermissionSetWithProfileIdList =
      [SELECT id,Name,IsOwnedByProfile,Profile.Name,
       ProfileId FROM PermissionSet where IsOwnedByProfile = true
       AND Id IN :PermissionSetList ];

    //Get the list of profiles associated with those permission sets
    Set<ID> ProfileList = new   Set<ID> ();
    for(PermissionSet per : PermissionSetWithProfileIdList){
       ProfileList.add(per.ProfileId);
       }
    //Get the list of users who have those profiles assigned
    List<User> UserList  =[SELECT id FROM User where ProfileId  IN :ProfileList ];


    //Assign the permission set in the new package to those users
    List<PermissionSet> PermissionSetToAssignList = [SELECT id,Name
      FROM PermissionSet where Name='TestPermSet' AND
      NamespacePrefix = 'TestPackage'];
    PermissionSet PermissionSetToAssign = PermissionSetToAssignList[0];

    Set<ID> UsersSet = new Set<ID> ();
    for(User us : UserList){
    PermissionSetAssignment psa= new PermissionSetAssignment();
    psa.PermissionSetId = PermissionSetToAssign.id;
    psa.AssigneeId = us.id;
    UsersSet.add(us.id);
    }
  }
}
```

```
// Test for the post install class
@isTest
private class PostInstallClassTest {
    @isTest
    public static void test() {
      PostInstallClass myClass = new PostInstallClass();
      Test.testInstall(myClass, null);
    }
}
```

Scheduling Push Upgrades

Note: Patch versions and push upgrades are only available to Salesforce ISV partners.

After you've created a patch version of your package, you can automatically deploy it to customers using a push upgrade.

Tip: Salesforce strongly recommends following this sequence for pushing package upgrades.

1. Push the upgrade to your own organizations so you can run tests and fix any bugs before upgrading subscribers.
2. When you're ready and have coordinated with your customers on their change management process, push to a small number of customer organizations. Try sandbox organizations first, if possible.
3. Once you're comfortable with the initial results, push to your wider customer base, based on your agreements with each customer.
4. Deprecate the previous version of your package in your main development organization. Replace the version on AppExchange if necessary, and update your Trialforce setup.
5. If your upgrade was a patch, after you've successfully distributed the upgrade to subscriber organizations, reintegrate those changes into your main development organization. For more information about combining patches in the main development organization, see Working with Patch Versions on page 277.

Read Best Practices for Push Upgrades and Patch Versions for more information.

EDITIONS

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

Available in: **Developer** Edition

USER PERMISSIONS

To push an upgrade:
- Upload AppExchange Packages

Schedule a Push Upgrade Using the UI

1. Log in to your main development org (not the patch org you used to upload the new version).
2. From Setup, enter `Packages` in the `Quick Find` box, then select **Packages**.
3. Click the name of the managed package whose upgrade you want to push.
4. On the package detail page, click the **Versions** tab, and then click **Push Upgrades**.
5. Click **Schedule Push Upgrades**.
6. Select a package version to push from the **Patch Version** drop-down list.

   Note: Beta versions aren't eligible for push.

7. For the scheduled start date, enter when you want the push upgrade to begin.
8. In the Select Target Organizations section, select the orgs to receive your push upgrade. If an org already received a push upgrade for the selected package version, it doesn't appear in this list. You can select orgs by:
   - Entering a term that filters based on an org's name or ID. Names can match by partial string, but IDs must match exactly.
   - Choosing between production and sandbox orgs from the **Organizations** dropdown list.
   - Choosing orgs that have already installed a particular version.
   - Clicking on individual orgs or the **Select All** and **Deselect All** checkboxes.

   This section lists the following information about the org (in alphabetical order):

Field	Description
`Current Version`	The current package version an organization has installed.

Field	Description
`Organization ID`	The ID that uniquely identifies the organization to Salesforce.
`Organization Name`	The name of the organization. Clicking this name shows the upgrade history for the organization.
`Primary Contact`	The name of the contact who installed the package.

9. Click **Schedule**. While a push upgrade is in progress, you can click **Abort** to stop it.

Schedule a Push Upgrade Using the Enterprise API

1. Authenticate to your main development org (not the patch org you used to upload the new version) according to the tool you're using.
2. Determine the package version you want to upgrade subscribers to by querying the MetadataPackageVersion object.
3. Gather the list of subscriber orgs that are eligible to be upgraded by querying the PackageSubscriber object.

   Note: If you are retrieving more than 2,000 subscribers, use the SOAP API`queryMore()` call.

4. Create a PackagePushRequest object. PackagePushRequest objects take a PackageVersionId and, optionally, a ScheduledStartTime parameter to specify when the push begins. If you omit the ScheduledStartTime, the push begins when you set the PackagePushRequest's status to `Pending`.
5. Create a PackagePushJob for each eligible subscriber and associate it with the PackagePushRequest you created in the previous step.
6. Schedule the push upgrade by changing the status of the PackagePushRequest to `Pending`.
7. Check the status of the PackagePushRequest and PackagePushJob objects by querying the `Status` fields. If the status is either Created or Pending, you can abort the push upgrade by changing the status of the PackagePushRequest to Canceled. You cannot abort a push upgrade that has a status of Canceled, Succeeded, Failed, or In Progress.

   Note: If you are pushing the upgrade to more than 2,000 subscribers, use the Bulk_API to process the job in batches.

For sample code and more details, see the object descriptions in the *Object Reference for Salesforce and Lightning Platform* or the *SOAP API Developer Guide*.

View Push Upgrade Details

Note: Patch versions and push upgrades are only available to Salesforce ISV partners.

For information about a specific push upgrade that your organization sent, from Setup, enter *Packages* in the Quick Find box, then select **Packages**. Click the name of the package that you want to view, and then click **Push Upgrades**. Clicking the name of a Target takes you to the Push Upgrade Details page, which has information for the push job and each organization that it was pushed to.

The Job Details section has the following information about the overall push upgrade (in alphabetical order):

Field	Description
End Date	The date and time the push upgrade finished.
Ignore Apex Test Failures	Whether Apex test failures that may cause the installed application not to function properly were ignored.
Scheduled By	The name of the user who initiated the push upgrade.
Start Date	The scheduled start date and time of the push upgrade.
Status	The status of the push upgrade, whether scheduled, in progress, completed, aborted, or completed with failures.
Version	The package version number that was pushed.

EDITIONS

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

Available in: **Developer** Edition

USER PERMISSIONS

To view push upgrade details:

- Upload AppExchange Packages

In the Organizations section, you can get a list of all the organizations that received a push upgrade. You can filter organizations by using the search box and entering a term that filters based on an org's name or ID. Names can match by partial string, but IDs must match exactly. From the drop-down list, you can also filter based on the status of the push upgrade.

The list contains the following information specific to each organization (in alphabetical order):

Field	Description
Duration	The amount of time the push upgrade took.
Failure Type	Lists the type of failure that occurred (if any). If the push upgrade did fail, a possible explanation is provided in the collapsible section. If the push upgrade was unsuccessful, click **Retry** to try it again.
Organization ID	The ID that uniquely identifies the organization to Salesforce.
Organization Name	The name of the organization. Clicking this name shows the upgrade history for the organization.
Start	The scheduled start date and time of the push upgrade.

Field	Description
Status	The status of the push upgrade, whether scheduled, in progress, completed, aborted, or completed with failures.

View an Organization's Upgrade History

Note: Patch versions and push upgrades are only available to Salesforce ISV partners.

For more information about a specific organization that received a push upgrade, from Setup, enter *Packages* in the Quick Find box, then select **Packages**. Click the name of the package that you want to view, and then click the name of a Target. Clicking an organization in the target list provides the following details (in alphabetical order):

Field	Description
Current Version	The current package version an organization has installed.
Organization ID	The ID that uniquely identifies the organization to Salesforce.
Organization Name	The name of the organization.
Primary Contact	The name of the contact who installed the package.
Primary Contact Email	The email address of the package publisher.
Status	The status of the push upgrade, whether scheduled, in progress, completed, aborted, or completed with failures.

EDITIONS

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

Available in: **Developer** Edition

USER PERMISSIONS

To view push upgrade history:

- Upload AppExchange Packages

The Push Upgrade History lists the following information (in alphabetical order):

Field	Description
Action	Clicking **View Details** returns you to the job details for that upgrade.
Start Date	The scheduled start date and time of the push upgrade.
Status	The status of the push upgrade, whether scheduled, in progress, completed, aborted, or completed with failures.
Version	The package version number that was pushed.

APPENDICES

APPENDIX A ISVforce User License Comparison

Introduction

The following tables compare object access, user permissions and features, and organization limits for these license types.

- Lightning Platform Administrator—A standard Salesforce license with complete customization capabilities. It prohibits Create, Read, Update, and Delete on Leads, Opportunities, Products, Cases, Solutions, and Campaigns.
- Lightning Platform—A standard Salesforce Platform license with access to Accounts, Contacts, and custom objects. Used by non-administrators.

Note: For a complete list of license types, see: https://help.salesforce.com/HTViewHelpDoc?id=users_license_types_available.htm

The following symbols are used in the tables.

- ✓ —Included in license
- $—Available as an add-on for an additional fee
- C—Create access to the object
- R—Read access to the object
- U—Update access to the object
- D—Delete access to the object

Object Accessed

Object Accessed	Lightning Platform Administrator		Lightning Platform	
	EE	UE/PXE	EE	UE/PXE
Accounts	CRUD	CRUD	CRUD	CRUD
Activities, Tasks	CRUD	CRUD	CRUD	CRUD
Assets				
Calendar, Events	CRUD	CRUD	CRUD	CRUD
Campaigns				
Cases				
Contacts	CRUD	CRUD	CRUD	CRUD
Content	CRUD	CRUD	CRUD	CRUD

Object Accessed	Lightning Platform Administrator		Lightning Platform	
	EE	UE/PXE	EE	UE/PXE
Contracts				
Documents	CRUD	CRUD	CRUD	CRUD
Entitlements				
Ideas	CRUD	CRUD	CR	CR
Knowledge	R	R		
Leads				
Opportunities				
Products & Price Books				
Questions and Answers	CRUD	CRUD		
Quotes				
Service Contracts				
Solutions				

User Features

User Features	Lightning Platform Administrator		Lightning Platform	
	EE	UE/PXE	EE	UE/PXE
Company Community	$	$	$	$
Content	✓	✓	✓	✓
Flows	✓	✓	✓	✓
Jigsaw Exports	$	$	$	$
Knowledge	$	$	$	$
Mobile (Full)	$	✓	$	✓
Offline	✓	✓	✓	✓
Send Mass Email	✓	✓	✓	✓
Siteforce Contributor	$	$	$	$
Siteforce Publisher	$	$	$	$

User Permissions

User Permissions	Lightning Platform Administrator		Lightning Platform	
	EE	UE/PXE	EE	UE/PXE
Customize Reports	✓	✓	✓	✓
Customize Dashboards	✓	✓	✓	✓
View Dashboards*	✓	✓	✓	✓
Chatter (Groups, Files, Profiles)	✓	✓	✓	✓
Create Workflow and Approval Process	✓	✓		
Manage Users and Profiles	✓	✓		
Identity	✓	✓	✓	✓
Identity Connect	$	$	$	$
Write Apex Code	✓	✓		
Work.com	$	$	$	$
Custom Apps Limit	10	Unlimited	10	Unlimited
Custom Tabs Limit	25	Unlimited	25	Unlimited
Custom Objects Limit**	200	2,000	200	2,000

* The running user of a dashboard must be a Lightning Platform or a Lightning Platform One App user to view the dashboard. Dashboards using the Lightning Platform administrator as the running user are not viewable by other Lightning Platform license types.

** Restricted limit for Lightning Platform One App and Chatter Plus.

Additional Organization Limits

Additional Organization Limits (Added Per User)	Lightning Platform Administrator		Lightning Platform	
	EE	UE/PXE	EE	UE/PXE
Data Storage	20 MB	120 MB	20 MB	120 MB
File Storage	2 GB	2 GB	2 GB	2 GB
API Calls (Per Day Per User)	1,000	5,000	1,000	5,000

For data storage, orgs of all editions are automatically allocated 10 GB. For each user added to an org, an extra 20 MB of storage is allocated, though Unlimited Edition and Performance Edition orgs receive more storage capacity. For each user added to an Unlimited Edition or Performance Edition org, an extra 120 MB of storage is allocated.

For file storage, Enterprise, Performance, and Unlimited Editions are allocated a per-user limit multiplied by the number of users in the organization plus an additional per-organization allocation of 11 GB. For example, an Enterprise Edition organization with 600 users receives 1,211 GB of file storage, or 2 GB per user multiplied by 600 users plus an additional 11 GB.

Note: For a complete list of storage limits for each edition, see: https://help.salesforce.com/HTViewHelpDoc?id=limits_storage_allocation.htm

APPENDIX B OEM User License Comparison

Compare object access, user permissions and features, and org limits for the license types available to partners.

License Types and Availability

Note: Partner Community licenses are no longer available for resale. If you're a partner with a customer who requires similar features, consider a Customer Community Plus license instead.

The following licenses are available to new and existing ISV partners.

- OEM Embedded—A full Lightning Platform license with contractual restrictions. It prohibits Create, Read, Update, and Delete on Leads, Opportunities, Cases, Solutions, and Campaigns.
- Customer Community—Similar to a High Volume Customer Portal license. Well suited for business-to-consumer communities with many external users.*
- Customer Community Plus—Similar to the Customer Community license, but adds more storage and access to features like Roles and Sharing.*

The following licenses aren't available to new partners, but can be resold by existing partners where noted.

- Partner Community—Similar to a Gold Partner license. Well suited for business-to-business communities, such as a partner community. Existing partners who currently sell Partner Community licenses can continue offering them.*
- ISV Portal—An Authenticated Website license with basic data sharing options (manual sharing to user and participation in sharing groups is not permitted). Users can only log in via Lightning Platform Sites. This is best used when projected user volumes will exceed 100,000. This legacy license type is no longer available.*
- ISV Portal with Sharing—A Customer Portal Manage Custom license with full sharing capabilities. Users can log in only via Lightning Platform Sites. Best used when projected user volumes are under 100,000 and granular security access is required. This legacy license type is no longer available.*

Licenses sold by partners can only be used to access the partner's app. End users can't develop or extend apps by creating custom objects, but they can access more apps as long as those apps are sold with an embedded license.

* Licenses can be assigned to external users only.

The following symbols are used in the tables:

- ✓ —Included in license
- $—Available as an add-on for a fee
- C—Create access to the object
- R—Read access to the object
- U—Update access to the object
- D—Delete access to the object

Objects

Object Accessed	OEM Embedded	ISV Portal	ISV Portal with Sharing	Customer Community	Customer Community Plus	Partner Community
Accounts	CRUD		CRU	R	CRU	CRUD
Activities, Tasks	CRUD			R	CRU	CRUD
Calendar, Events	CRUD					CRUD
Contacts	CRUD		CRU	R	CRU	CRUD
Content	CRUD		R		View and Upload	CRUD
Contracts*	CRUD	CRU	CRU	CRUD	CRUD	CRUD
Documents	CRUD	R	R	R	R	R
Ideas	CR	CR	CR	CR	CR	CR
Orders*	CRUD	CRU	CRU	CRUD	CRUD	CRUD
Products & Price Books*	CRUD	CRU	CRU	R	R	R
ISV Custom Object	CRUD	CRUD	CRUD	CRUD	CRUD	CRUD

*The Orders permission set license (PSL) enables administrators to grant users with Lightning Platform user licenses access to Contracts, Products, Price Books, and Orders. The Orders Platform PSL is available to OEM partners only. Orders functionality is automatically available to all licenses except Lightning Platform licenses, which explicitly require the new PSL to grant access.

User Features

User Feature	OEM Embedded	ISV Portal	ISV Portal with Sharing	Customer Community	Customer Community Plus	Partner Community
Knowledge	$		R	R	R	R
Send Mass Email	✓					
Salesforce Mobile App	✓	✓	✓	✓	✓	✓
Identity	✓	✓	✓	✓	✓	✓
Flows	✓	✓	✓	✓	✓	✓
Territory Management	✓		✓			✓

User Permissions

User Permission	OEM Embedded	ISV Portal	ISV Portal with Sharing	Customer Community	Customer Community Plus	Partner Community
Create and Customize Reports	✓				Create and Manage	Create and Manage
View Reports	✓		✓		✓	✓
Create and Customize Dashboards	✓				Create and Manage	
View Dashboards*	✓				✓	✓
Enhanced User/Role Based Sharing	✓		✓		✓	✓
Identity	✓	✓	✓	✓	✓	✓
Identity Connect					✓	
Chatter (Groups, Files, Profiles)	✓			✓	✓	✓
Submit Workflow Approvals	✓		✓	✓	✓	✓
Custom Apps Limit	1					
Custom Tabs Limit	25	25	25	25		25
Custom Objects Limit	400**	200	200	200		200

* The running user of a dashboard must be a Lightning Platform user to view the dashboard. Dashboards using the Lightning Platform administrator as the running user are not viewable by other Lightning Platform license types.

** The limit of 400 custom objects applies to the primary app offering. Subscribers cannot create their own custom objects.

Storage Limits

Additional Org Limits (Added Per User)	OEM Embedded	Customer Community	Customer Community Plus	Partner Community
Data Storage	20 MB	0	2 MB per member (member-based license) 1 MB per member	5 MB

Additional Org Limits (Added Per User)	OEM Embedded	Customer Community	Customer Community Plus	Partner Community
			(login-based license)	
File Storage	2 GB	0	0	0

For data storage, each OEM Embedded org is automatically allocated 10 GB. For each added user, an extra 20 MB of storage is allocated.

For file storage, each OEM Embedded org is allocated a per-user limit multiplied by the number of users in the org plus a per org allocation of 11 GB. For example, an OEM Embedded org with 600 users receives 1,211 GB of file storage, or 2 GB per user multiplied by 600 users plus 11 GB.

Salesforce Edition	Data Storage Minimum per Org	File Storage Minimum per Org	Storage Allocation per User License
OEM Embedded	10 GB	11 GB	20 MB of data storage and 2 GB of file storage

API Limits

The following table lists the limits for the total API requests (calls) for an OEM Embedded org.

Salesforce Edition	API Calls per License Type	Total Calls per 24-Hour Period
OEM Embedded	• Salesforce: 1,000 • Salesforce Platform: 1,000	15,000 + (number of licenses x calls per license type), up to a maximum of 1,000,000

Limits are enforced against the aggregate of all API calls made by the org in a 24 hour period. Limits are not on a per-user basis. When an org exceeds a limit, all users in the org can be temporarily blocked from making additional calls. Calls are blocked until usage for the preceding 24 hours drops below the limit.

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.

INDEX

A

Apex
- behavior in packages 279
- deprecation effects 280
- Metadata access 57

API token
- requesting 181

app
- installation options 151
- patch (version) updates 180
- publish 148
- search optimization 180
- sell 148
- submit for security approval 152

AppExchange
- deleting components 283

AppExchange Checkout 175, 183

Apps
- uploading 28, 120

Attributes 25

B

Beta packages
- uninstalling 128
- uploading 118

C

Channel Order App 184

Checkout 175, 183

company name 179

Components 25

connect to Dev Hub org 112

Creating packages 28, 120

Creating patches 276

D

data file 268

Dependencies 56

Deployment 128

dev hub 111

Developer Tools 111–112

Developing
- partner WSDL 67

Dynamic Apex
- supporting multiple editions 75

E

Editions 3

Extending packages 73

F

Feature Management App
- considerations 247
- installation 242
- known issues 247
- setup 242

Feature parameters
- best practices 246
- considerations 247
- de-protecting custom objects 246
- de-protecting custom permissions 246
- fields 241
- known issues 247
- limits 246
- LMO-to-subscriber 244–245
- objects 241
- protecting custom permissions 246
- subscriber-to-LMO 245
- types 241

feedback 182

FMA 240–242, 244–247

free trial
- create 142, 182
- vs test drive 182

G

Group edition
- access control 71
- limits 71
- packages 73
- using Apex 71

I

ideas 182

industries 179

Installing packages 125

introduction 267

L

leads
- source codes 175–177

leads *(continued)*
vs license records 176
licenses
choose settings 152
vs lead settings 176
listing
add a test drive 174
add categories 153
analytics 177
delisted by Salesforce 182
submit for security approval 152
Login 265

M

manage scratch orgs 111
managed package
change 180
change a listing 181
register 152
Managed packages
component availability 128
component behavior 46
group edition 71, 73
limits for group edition 71
limits for professional edition 71
Metadata in Apex code 57
planning 26
professional edition 71, 73
push upgrades 286, 288
status 25
supporting multiple editions 73, 75
upgrading 274
Metadata access in Apex 57
MetricsDataFile object 269

O

Objects
MetricsDataFile 269
Operational scope 56

P

Package installation 125
Package versions
behavior versioning Apex 279
deprecating Apex 280
Packages
about 2, 25
component availability 128
component behavior 46
Packages *(continued)*
creating 28, 120
deleting components 283
dependencies 56
designing 24
developing 25
distributing 146
Editions 25
installing packages 75
installing using the API 129
managed 25
managing feature access 240–247
status 25
terminology 25
tracking activation metrics 240–247
understanding 25
uninstalling using the API 129
unmanaged 25
uploading 28, 120
partner account 179
Partner WSDL 67
Patch versions
creating 276
uploading 276
patches 180
Permission sets 58
popularity 181
Professional edition
access control 71
limits 71
packages 73
using Apex 71
Profile settings 58
provider profile
and partner accounts 179
create or edit 149
publish
create a test drive 174
making your listing public 148
Publishing console 149
Push upgrades 286, 288

R

register namespace 112
relevance 181
resources 141
review
edit 180
write 182

S

Salesforce DX 111–112
sandbox 180
scratch org allocations 108
search optimization 180
security review
    requirements 152
    submit apps and trial templates 152
Security review 142–145
services
    publish 148
    search optimization 180
    sell 148
setting up 267
Subscriber support 265
Supporting multiple editions 75

T

test drive
    create or edit 174
    vs free trial 182
Testing 3, 115
trial template
    description 142, 182
    submit for security approval 152
Trialforce
    Lightning Experience 262
Tutorials 4

U

Uninstalling packages 128
Unmanaged packages
    component behavior 46
    Metadata in Apex code 57
Upgrading packages 274
Uploading beta packages 118
Uploading packages 28, 120
Uploading patches 276
Usage Metrics 267–268
using an extension package 73
using dynamic Apex 73, 75