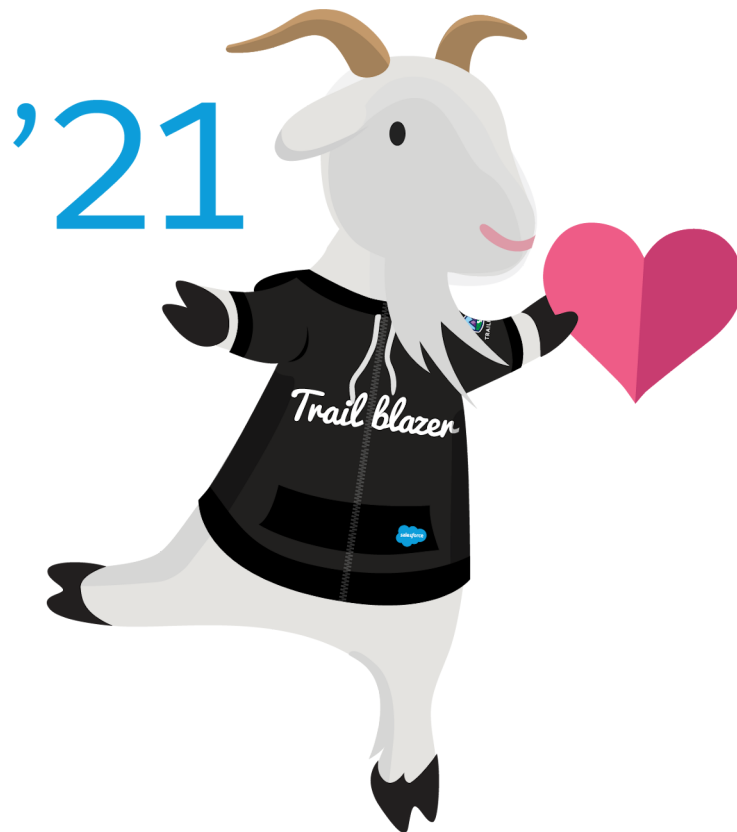


---

# ISVforce Guide

Version 52.0, Summer '21





# CONTENTS

<b>Chapter 1: Welcome to the ISVforce Guide</b>	<b>1</b>
Resources for Partners	2
Roles in the Solution Lifecycle	2
How to Sign Up for Test Environments	3
<b>Chapter 2: ISVforce Quick Start</b>	<b>4</b>
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	13
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
<b>Chapter 3: Grow Your AppExchange Business</b>	<b>17</b>
<b>Chapter 4: Design and Build Your Solution</b>	<b>18</b>
Create a Secure Solution	19
Prevent Common Violations of Secure Coding Guidelines	20
Loading JavaScript Files from Third-Party Endpoints	21
Loading Third-Party CSS in Lightning Components	22
Using CSS Outside Components	23

## Contents

Running JavaScript in the Salesforce Domain . . . . .	23
Exposing Secret Data When Debugging . . . . .	24
Storing Sensitive Data Insecurely . . . . .	25
Using Software That Has Known Vulnerabilities . . . . .	25
Using Sample Code in Production . . . . .	26
Bypassing Object-Level and Field-Level Access Settings . . . . .	26
Bypassing Sharing Rules in Apex . . . . .	27
SOQL Injection Due to Insecure Database Query Construction . . . . .	28
Cross-Site Request Forgery . . . . .	29
Open Redirects . . . . .	30
Lightning LockerService Disabled . . . . .	31
Insufficient Escaping in Lightning Components . . . . .	32
Asynchronous Code in Components . . . . .	33
Overview of Packages . . . . .	34
Planning the Release of Managed Packages . . . . .	35
Create a Package . . . . .	36
Developing and Distributing Unmanaged Packages . . . . .	37
Create and Upload an Unmanaged Package . . . . .	37
Install Notifications for Unauthorized Managed Packages . . . . .	38
Components Available in Managed Packages . . . . .	38
Components Available in Unmanaged Packages . . . . .	44
Editing Components and Attributes After Installation . . . . .	47
Components Automatically Added to Packages . . . . .	55
Special Behavior of Components in Packages . . . . .	58
Protected Components . . . . .	70
Set Up a Platform Cache Partition with Provider Free Capacity . . . . .	71
Understanding Dependencies . . . . .	72
Metadata Access in Apex Code . . . . .	73
About Permission Sets and Profile Settings . . . . .	73
Permission Set Groups . . . . .	75
Custom Profile Settings . . . . .	75
Protecting Your Intellectual Property . . . . .	76
Creating Packaged Applications with Chatter . . . . .	76
Matching the Salesforce Look and Feel . . . . .	77
Maintaining My Domain and Visualforce URLs . . . . .	78
Developing App Documentation . . . . .	78
About API and Dynamic Apex Access in Packages . . . . .	79
Manage API and Dynamic Apex Access in Packages . . . . .	81
Configuring Default Package Versions for API Calls . . . . .	82
About the Partner WSDL . . . . .	83
Generating an Enterprise WSDL with Managed Packages . . . . .	84
Work with Services Outside of Salesforce . . . . .	84
Architectural Considerations for Group and Professional Editions . . . . .	85
Features in Group and Professional Editions . . . . .	86

## Contents

Limits for Group and Professional Editions . . . . .	87
Access Control in Group and Professional Editions . . . . .	87
Using Apex in Group and Professional Editions . . . . .	87
API Access in Group and Professional Editions . . . . .	88
Designing Your App to Support Multiple Editions . . . . .	89
Sample Design Scenarios for Group and Professional Editions . . . . .	91
Connected Apps . . . . .	91
Environment Hub . . . . .	92
Get Started with the Environment Hub . . . . .	93
Manage Orgs in the Environment Hub . . . . .	95
Single Sign-on in the Environment Hub . . . . .	97
Environment Hub Best Practices . . . . .	99
Environment Hub FAQ . . . . .	100
Considerations for the Environment Hub in Lightning Experience . . . . .	102
Developer Hub . . . . .	102
Scratch Org Allocations for Partners . . . . .	103
Enable Dev Hub Features in Your Org . . . . .	104
Add Salesforce DX Users . . . . .	104
Free Limited Access License . . . . .	105
Manage Scratch Orgs from Dev Hub . . . . .	105
Link a Namespace to a Dev Hub Org . . . . .	106
Supported Scratch Org Editions for Partners . . . . .	107
Notifications for Package Errors . . . . .	107
Set the Notification Email Address . . . . .	108
<b>Chapter 5: Package and Test Your Solution . . . . .</b>	<b>109</b>
About Managed Packages . . . . .	110
Configure Your Developer Settings . . . . .	111
Register a Namespace Prefix . . . . .	111
Specifying a License Management Organization . . . . .	112
What are Beta Versions of Managed Packages? . . . . .	112
Creating and Uploading a Beta Package . . . . .	113
Create and Upload a Managed Package . . . . .	115
View Package Details . . . . .	117
Installing a Package . . . . .	120
Component Availability After Deployment . . . . .	123
Uninstalling a Package . . . . .	123
Installing Managed Packages Using the API . . . . .	124
Resolving Apex Test Failures . . . . .	125
Running Apex on Package Install/Upgrade . . . . .	125
How does a Post Install Script Work? . . . . .	126
Example of a Post Install Script . . . . .	127
Specifying a Post Install Script . . . . .	129
Running Apex on Package Uninstall . . . . .	129

How does an Uninstall Script Work? .....	129
Example of an Uninstall Script .....	130
Specifying an Uninstall Script .....	131
Publishing Extensions to Managed Packages .....	131
<b>Chapter 6: Pass the AppExchange Security Review .....</b>	<b>132</b>
AppExchange Security Review .....	133
How Does AppExchange Security Review Work? .....	133
Partner Security Portal .....	136
Set Up Your Partner Security Portal Login .....	136
Security Scanners on the Portal .....	137
Office Hours Appointments on the Portal .....	138
Test Your Entire Solution .....	139
False Positives .....	141
Document Your Responses to False Positives .....	142
Example Responses to False Positives in Checkmarx Scan Results .....	142
Example Responses to False Positives in a Security Review Failure Report .....	143
Security Review Resources .....	144
<b>Chapter 7: Publish Your Solution on AppExchange .....</b>	<b>145</b>
What Is AppExchange? .....	146
Business Plans for AppExchange Listings .....	146
Submit a Due Diligence and Compliance Certification .....	147
Publish on AppExchange .....	148
Connect a Packaging Org to the Publishing Console .....	149
Create or Edit Your Provider Profile .....	149
Create or Edit Your AppExchange Listing .....	149
Add a Business Plan to an AppExchange Listing .....	150
Make Your AppExchange Listing Effective .....	151
Select an Installation Option .....	151
Register Your Package and Choose License Settings .....	152
Complete the Security Review Cycle .....	152
How Does AppExchange Search Work? .....	163
Email Notifications .....	166
Collect AppExchange Leads .....	167
AppExchange Leads .....	167
AppExchange Leads and License Activities .....	168
Enable AppExchange Lead Collection .....	168
AppExchange Lead Source Codes .....	168
Troubleshoot AppExchange Leads .....	169
Analytics Reports for Publishers .....	172
Update the Package in Your AppExchange Listing .....	173
AppExchange FAQ .....	174
Can I add more industries? .....	174

Do I need an APO to publish my app or component on the AppExchange? . . . . .	174
Can I change my company name? . . . . .	174
Can I create my app or component on a Salesforce sandbox and upload it to the AppExchange? . . . . .	175
Can I edit a review? . . . . .	175
Can I keep the same listing but change the package it provides? . . . . .	175
Can I Update My Solution with a New Version or Patch? . . . . .	175
How Do Customers Find My Listing? . . . . .	175
How do I edit a package after I've created a listing? . . . . .	176
How do I get an API token for my app? . . . . .	176
How do I increase my listing's popularity? . . . . .	176
How do I offer a free trial of my app or component? . . . . .	177
How do I see listings that Salesforce removed? . . . . .	177
How do I upgrade my customers to a new version? . . . . .	177
What's the difference between a free trial and test drive? . . . . .	177
Where can I share my ideas? . . . . .	177
Where can I write a review? . . . . .	178
Can I have multiple listings for an app or component? . . . . .	178
<b>Chapter 8: Sell on AppExchange with Checkout . . . . .</b>	<b>179</b>
AppExchange Checkout . . . . .	180
How Is Revenue Shared in AppExchange Checkout? . . . . .	181
Payment Plans in AppExchange Checkout . . . . .	181
Payment Methods in AppExchange Checkout . . . . .	182
Get Started with AppExchange Checkout . . . . .	182
Support International Payments in AppExchange Checkout . . . . .	188
Manage AppExchange Checkout Subscriptions . . . . .	193
AppExchange Checkout FAQs . . . . .	194
AppExchange Checkout Considerations . . . . .	197
Checkout Management App . . . . .	197
Checkout Management App Best Practices . . . . .	199
Checkout Management App Objects . . . . .	199
Get Started with the Checkout Management App . . . . .	201
Sample Checkout Management App Customizations . . . . .	206
Update Settings in the Checkout Management App . . . . .	208
View Checkout Management App Logs . . . . .	209
<b>Chapter 9: Monitor Performance with Analytics for AppExchange Partners . . . . .</b>	<b>210</b>
Monitor Your AppExchange Performance with Marketplace Analytics . . . . .	211
Marketplace Analytics Dashboard . . . . .	211
Get Started with the Marketplace Analytics Dashboard . . . . .	231
Marketplace Analytics FAQs . . . . .	233
AppExchange App Analytics . . . . .	235
Request AppExchange App Analytics . . . . .	236

Download Package Log Files, Usage Summaries, and Subscriber Snapshots . . . . .	236
AppExchange App Analytics Best Practices . . . . .	237
Package Usage Summaries . . . . .	253
Package Log Files . . . . .	255
Subscriber Snapshots . . . . .	262
Test Custom Integrations . . . . .	263
<b>Chapter 10: Manage Orders . . . . .</b>	<b>265</b>
Channel Order App . . . . .	266
Channel Order App Objects . . . . .	266
Order Types . . . . .	268
Order Status . . . . .	269
Channel Order App Permission Sets . . . . .	270
Set Up the Channel Order App . . . . .	271
Install the Channel Order App . . . . .	271
Assign Permission Sets to Channel Order App Users . . . . .	272
Define a Channel Order App Email Service . . . . .	272
Connect the Channel Order App to Salesforce . . . . .	273
Display Customers in the Channel Order App . . . . .	274
Assign Page Layouts in the Channel Order App . . . . .	274
Upgrade the Channel Order App . . . . .	275
Channel Order App Upgrade Considerations . . . . .	275
Upgrade the Channel Order App . . . . .	276
Field Mapping in Channel Order App v2 and Later . . . . .	278
Manage Orders in the Channel Order App . . . . .	280
Submit an Order . . . . .	280
Edit an Order . . . . .	282
Clone an Order . . . . .	283
Recall an Order . . . . .	283
Delete a Draft Order . . . . .	283
Fix Errors on Returned Orders . . . . .	284
Channel Order Apex API . . . . .	284
CHANNEL_ORDERS Namespace . . . . .	284
Service Order . . . . .	299
Service Order Detail . . . . .	301
Partner Order Submit API . . . . .	304
<b>Chapter 11: Managing Licenses for Managed Packages . . . . .</b>	<b>307</b>
Get Started with the License Management App . . . . .	308
Install the License Management App . . . . .	308
Associate a Package with the License Management App . . . . .	309
Configure Permissions for the License Management App . . . . .	309
Lead and License Records in the License Management App . . . . .	311
Modify a License Record . . . . .	311

## Contents

Refresh Licenses for a Managed Package .....	312
Extending the License Management App .....	312
Package and Package Version Object Fields .....	313
License Object Fields .....	313
Adding Custom Automation to License Management App Objects .....	315
Move the License Management App to Another Salesforce Org .....	315
Troubleshoot the License Management App .....	316
Leads and Licenses Aren't Being Created in the License Management App .....	316
Proxy User Has Deactivated Message in the LMA .....	317
Best Practices for the License Management App .....	317
Troubleshoot Subscriber Issues .....	317
Request Login Access from Subscribers .....	318
Log In to Subscriber Orgs .....	318
Debug Subscriber Orgs .....	319
<b>Chapter 12: Manage Features</b> .....	<b>320</b>
Feature Parameter Metadata Types and Custom Objects .....	321
Set Up Feature Parameters .....	322
Install and Set Up the Feature Management App in Your License Management Org . . .	322
Create Feature Parameters in Your Packaging Org .....	323
Add Feature Parameters to Your Managed Package .....	323
Reference Feature Parameters to Drive App Behavior and Track Activation Metrics .....	324
How Do Feature Parameters Work? .....	324
Drive App Behavior with LMO-to-Subscriber Feature Parameters .....	324
Track Preferences and Activation Metrics with Subscriber-to-LMO Feature Parameters . .	325
Hide Custom Objects and Custom Permissions in Your Subscribers' Orgs .....	326
Best Practices for Feature Management .....	326
Considerations for Feature Management .....	327
<b>Chapter 13: Provide a Free Trial of Your Solution</b> .....	<b>328</b>
Why Use Trialforce? .....	329
Trialforce .....	329
Set Up Trialforce .....	331
Link a Package with Your License Management Organization .....	331
Request a Trialforce Management Org .....	332
Setting Up Custom Branding for Trialforce .....	332
Create a Trialforce Source Organization .....	334
Create a Trialforce Template .....	335
Link a Trialforce Template to the AppExchange .....	336
Submit a Trialforce Template for Security Review .....	336
Provide a Free Trial on the AppExchange .....	336
Provide a Free Trial on the AppExchange Using Trialforce .....	337
Offer a Test Drive on AppExchange .....	337
Provide a Free Trial on the AppExchange When Your Offering Is Installed .....	338

Provide Free Trials on Your Website . . . . .	338
Enable the SignupRequest API . . . . .	338
Choose a Sign-Up Form Hosting Option . . . . .	339
Create Sign-Ups Using the API . . . . .	340
Provision Trial Orgs . . . . .	340
Update Your Trial . . . . .	340
Trialforce Best Practices . . . . .	341
Trialforce FAQ . . . . .	341
How do I upgrade my trial with a new version of my offering? . . . . .	342
Can I distribute my app or component using both Trialforce and the AppExchange? . . . . .	342
How are trials different from Trialforce? . . . . .	342
Is it possible to install another app in a trial organization? . . . . .	342
How do I configure who can use Lightning Experience in my trial org? . . . . .	342
<b>Chapter 14: Support Your AppExchange Customers . . . . .</b>	<b>344</b>
Usage Metrics . . . . .	345
Setting up Usage Metrics . . . . .	345
Accessing Usage Metrics Data . . . . .	346
MetricsDataFile . . . . .	347
Usage Metrics Visualization . . . . .	349
<b>Chapter 15: Update Your Solution . . . . .</b>	<b>352</b>
About Package Versions . . . . .	353
Create and Upload Patches . . . . .	354
Working with Patch Versions . . . . .	355
Versioning Apex Code . . . . .	356
Apex Deprecation Effects for Subscribers . . . . .	357
Publish Upgrades to Managed Packages . . . . .	358
Delete Components from First-Generation Managed Packages . . . . .	359
Viewing Deleted Components . . . . .	361
Modifying Custom Fields after a Package is Released . . . . .	362
Manage Versions . . . . .	362
Push Package Upgrades to Subscribers . . . . .	362
Push Upgrades . . . . .	363
Push Upgrade Best Practices . . . . .	363
Assign Access to New and Changed Features . . . . .	364
Sample Post Install Script for a Push Upgrade . . . . .	365
Scheduling Push Upgrades . . . . .	366
<b>APPENDICES . . . . .</b>	<b>369</b>
<b>Appendix A: ISVforce User License Comparison . . . . .</b>	<b>369</b>
<b>Appendix B: OEM User License Comparison . . . . .</b>	<b>373</b>

**GLOSSARY** ..... 377

**INDEX** ..... 380



# CHAPTER 1 Welcome to the ISVforce Guide

## In this chapter ...

- [Resources for Partners](#)
- [Roles in the Solution 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.
- 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 Solution Lifecycle

---

This guide covers the entire lifecycle of a solution, so not all topics are relevant for every role. The following list has topic suggestions by role.

### **An application *architect***

The application architect determines the scope of the solution and the internal structures that support it. Architects need details about the underlying Lightning Platform platform that determine not only the solution's use, but which editions it supports, how it's installed, configured, and upgraded. We strongly recommend that architects review this entire guide, but especially the following chapters:

- [Design and Build Your Solution](#) on page 18
- [Pass the AppExchange Security Review](#) on page 132

### **A *developer* creates, packages, and uploads a solution**

A developer, or often a team of developers, creates a solution, packages it, and uploads it to AppExchange. Developers also update the solution with bug fixes and new features. As a developer, review the following chapters:

- [Design and Build Your Solution](#) on page 18
- [Package and Test Your Solution](#) on page 109
- [Developing App Documentation](#) on page 78
- [Update Your Solution](#) on page 352

### **A *publisher* distributes, sells, and supports the solution**

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

- [Publish Your Solution on AppExchange](#) on page 145
- [Provide a Free Trial of Your Solution](#) on page 328
- [Support Your AppExchange Customers](#) on page 344

### **An *administrator* installs the solution**

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

- [Installing a Package](#) on page 120

## 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 377.

## 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 you 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 85.

## 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 (PBO) provisioned for you. Your PBO 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, click **New Case**.
2. Hover over **Benefits & Tools**, and then click **Create a Case**.
3. From the Subtopic dropdown list, select **Partner Business Org**.
4. 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, provide the org ID to add two more CRM licenses to that 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 click **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.

5. 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 known 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

The screenshot shows the Visualforce Page Editor interface. At the top, the page is titled "Greeting". Below the title, there are "Page Edit" buttons: "Save", "Quick Save", and "Cancel". The "Page Information" section contains fields for "Label" (Greeting), "Name" (Greeting), and "Description". Below this, there are tabs for "Visualforce Markup" and "Version Settings". The "Visualforce Markup" tab is active, showing the following code:

```

1 <apex:page >
2 <!-- Begin Default Content REMOVE THIS -->
3 <h1>Hello World!</h1>
4 This is your new Page
5 <!-- End Default Content REMOVE THIS -->
6 </apex:page>

```

The line containing the `<h1>Hello World!</h1>` tag is highlighted with a red box.

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 you 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.

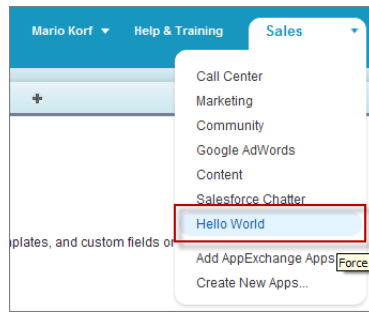
1. Click the Installation URL now.

### Installation URL Link

Package Version Detail		Change Password	Deprecate
Package Name	Hello World		
Version Name	Spring 2011		
Version Number	1.0 (Beta 1)		
Language	English		
Installation URL	<a href="https://login.salesforce.com/?startURL=%2Fpackaging%2FinstallPackage.apexp%3Fp0%3D04A0000000lxSk">https://login.salesforce.com/?startURL=%2Fpackaging%2FinstallPackage.apexp%3Fp0%3D04A0000000lxSk</a>		
Description		Change Password	Deprecate

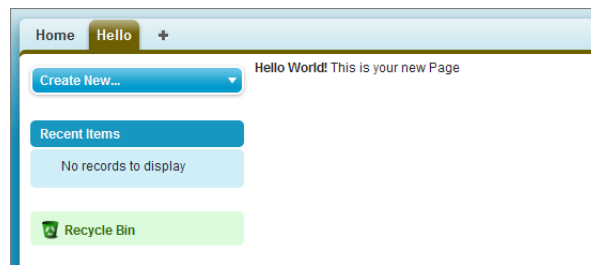
2. On the login page, enter the Username and Password of your test org.
3. On the Package Installation Details page, click **Continue**.
4. Click **Next**.
5. On the Security Level page, **Grant access to all users** and click **Next**.
6. Click **Install**.
7. Once the installation completes, you can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

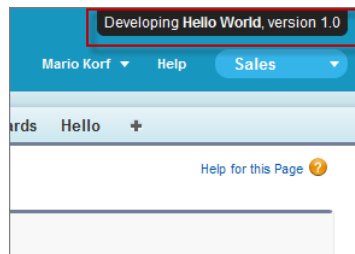
Imagine you've been through a few development cycles with your beta and you're ready to publish a public app. The next step is to upload a production app, or what we call a *managed released* version of your app. Then you can create a listing so that other people can find your app and know what it does.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

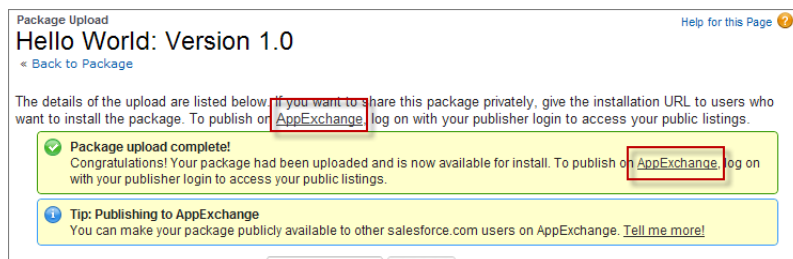


3. On the Package Detail page, click **Upload**.
4. For the Release Type, choose **Managed — Released**.
5. Scroll to the bottom and click **Upload**.
6. Click **OK** on the popup.

### 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](https://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 package to AppExchange and created a listing for your solution. You also linked your solution to the License Management App (LMA) available in your business org. You can use the LMA to manage and renew licenses and to set default license settings. For example, you can license your solution as a free trial that expires after a specified number of days. For more information, see [Manage Licenses for Managed Packages](#).

Right now your solution has a private listing on AppExchange. You can share the listing with potential customers, but the public doesn't see it unless they have the link. Before you can list the solution publicly, it must pass a security review, which is beyond the scope of this quick start. For more information, see [Pass the AppExchange Security Review](#) on page 132.

## 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.

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.

To learn how to push an upgrade to customers, see [Pushing an Upgrade](#).

## 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

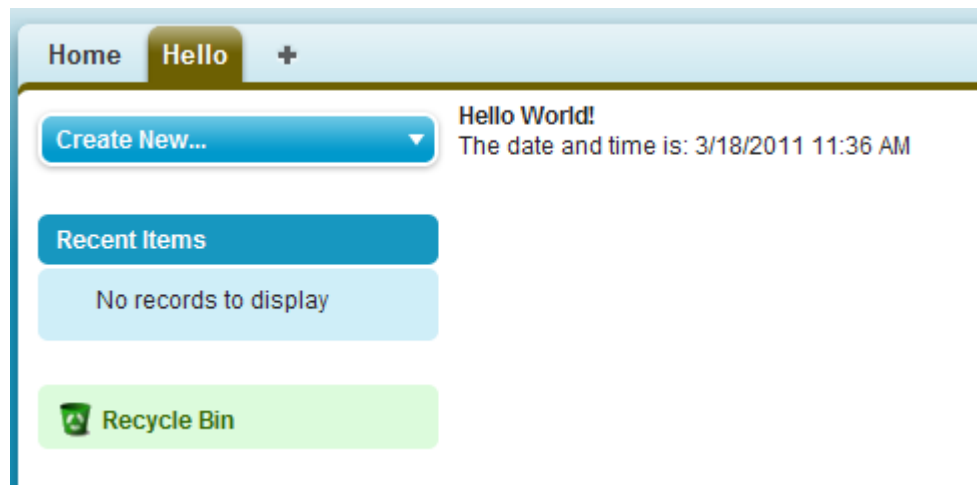
We're going to make a simple change to your app. Instead of displaying just Hello World, you'll add today's date.

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.

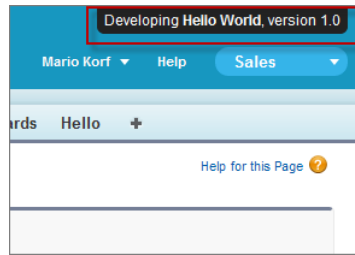
## Step 4: Installing or Pushing a Patch

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

Track your company's growth throughout the partner journey with the AppExchange Partner Trailblazer Score. Get a comprehensive view of your contributions across three pillars: customer success, innovation, and engagement. Explore detailed metrics within each pillar to see your company's progress as a Salesforce partner.

SEE ALSO:

[Salesforce AppExchange Partner Program](#)

## CHAPTER 4 Design and Build Your Solution

### In this chapter ...

- [Create a Secure Solution](#)
- [Prevent Common Violations of Secure Coding Guidelines](#)
- [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.

## Create a Secure Solution

---

Learn about practices and resources that help you develop a solution that resists common security threats. Designate a security expert on your development team. Review the AppExchange Security Requirements Checklist sections and Open Web Application Security Project (OWASP) guidelines that apply to your solution.

### Designate a Security Expert

Protecting your solution from security threats is easier when you integrate security considerations into all stages of development. One of the best ways to ensure that your solution follows security guidelines is to designate a security expert on your development team.

Have your entire development team collaborate with the security expert through all stages of development: design, implementation, and testing. Postponing security considerations until the final stages of development increases the likelihood that your team unknowingly propagates security violations as they code.

Regular collaboration prevents needless accumulation of security violations, and helps avoid delays in preparing a successful AppExchange security review submission.

### Learn How to Develop Secure Web Solutions

In [Prevent Common Violations of Secure Coding Guidelines](#) and in the [Secure Coding Guide](#), learn how to identify, prevent, and remediate security violations in solutions built on or integrated with the Salesforce Platform. Find out which violations most commonly appear, why they pose security risks, and how to avoid them in your code.

### Review the AppExchange Security Requirements

The AppExchange Security Requirements Checklist is our most comprehensive information resource for evaluating the security of your solution. To understand our baseline technical security requirements, review this checklist. As you develop your solution, meet the security requirements that apply to your code.

Be sure to parse through the entire [AppExchange Security Requirements Checklist](#). Requirements that apply to your solution can be spread out in various sections. However, the Best Practices for Security section applies to all solutions.



**Note:** In the checklist, *composite submission* refers to any solution that involves a third-party, non-Salesforce endpoint, system, or API.

### Follow Open Web Application Security Project Guidance

The Open Web Application Security Project (OWASP) website provides comprehensive information about web app security risks. It includes detailed guidance on how to test for, prevent, and resolve security issues. Familiarize yourself with the key resources on the OWASP website.

We recommend that you use the [OWASP Top Ten Project](#) as a primary reference for securing your solution. This section of the OWASP site documents the 10 most prominent security risks that appear in web apps. The guidelines are especially pertinent to web apps and services that aren't hosted on the Salesforce Platform.

Aim to follow development practices and security guidelines that conform as closely as possible to the [OWASP Secure Coding Practices - Quick Reference Guide](#).

SEE ALSO:

[Prevent Common Violations of Secure Coding Guidelines](#)

[Secure Coding Guide](#)

[AppExchange Security Review Trailhead module](#)

[Secure Coding Guide](#)

[OWASP site](#)

[OWASP Top Ten](#)

[OWASP Secure Coding Practices-Quick Reference Guide](#)

## Prevent Common Violations of Secure Coding Guidelines

---

The AppExchange security review assesses a solution's vulnerability to the most common security attacks. To pass the review, a solution can't violate secure coding guidelines. Only solutions that pass the review can be publicly distributed on AppExchange. Increase the likelihood that your solution passes. Learn which violations appear the most in reviewed solutions, why they pose security risks, and how to avoid them in your code.

### [Loading JavaScript Files from Third-Party Endpoints](#)

Avoid dynamically loading third-party JavaScript files from content delivery networks (CDNs). Instead, load the code from the static resources folder of your package.

### [Loading Third-Party CSS in Lightning Components](#)

Include cascading style sheets (CSS) and other resources in static resources rather than loading from a third party.

### [Using CSS Outside Components](#)

The Salesforce Platform tries to ensure that each namespace is an isolated sandbox, but isolation can't always be guaranteed. Where a namespace isolation breach occurs, one component can steal clicks from another component, or otherwise interfere with another component's intended use. To prevent this type of abuse, don't use CSS directives known to be incompatible with style isolation in your components.

### [Running JavaScript in the Salesforce Domain](#)

JavaScript code from multiple vendors can run in the same origin. To prevent code interference, vendor JavaScript code is sandboxed. Don't attempt to break out of a sandbox or run code outside your origin. Use Visualforce, Aura, or Lightning Web Components, which run in the proper origin.

### [Exposing Secret Data When Debugging](#)

In production environments, logging secret data with debug statements is a security vulnerability. Don't log secret data, sensitive information, passwords, keys, or stack traces in production environments. Redact the data or omit it from the logs.

### [Storing Sensitive Data Insecurely](#)

Follow enterprise security standards when you export data from the Salesforce Platform and when you store secret data in the platform.

### [Using Software That Has Known Vulnerabilities](#)

Using software that has documented common vulnerabilities and exposures (CVE) related to your use cases is a security vulnerability. If your solution has known vulnerabilities, test and deploy security patches as soon as they're available. If your solution uses software that has CVE-documented vulnerabilities unrelated to your use cases, prepare false positive documentation.

### Using Sample Code in Production

Only use sample code as an educational tool in preparation for developing your own application. When building your production code, always write the code yourself. Avoid copying code from sources that you don't directly control.

### Bypassing Object-Level and Field-Level Access Settings

Design your solutions to enforce the org's create, read, update, and delete (CRUD) and field-level security (FLS) settings on standard and custom objects.

### Bypassing Sharing Rules in Apex

Respect profile-based permissions, field-level security, sharing rules, and org-wide defaults in your Apex code.

### SOQL Injection Due to Insecure Database Query Construction

To prevent Salesforce Object Query Language (SOQL) injection, use bind variables and input sanitation.

### Cross-Site Request Forgery

A cross-site request forgery (CSRF) is an attack that forces an end user to execute unwanted actions during their authenticated web application session. To protect against CSRF, use `confirmationTokenRequired`, or trigger state changes with user actions.

### Open Redirects

An open redirect occurs when an application dynamically redirects to a user-controlled parameter value without any validation. Prevent open redirects by using hardcoded redirects.

### Lightning LockerService Disabled

Lightning LockerService is a critical security feature for Lightning code. It provides component isolation that allows code from many sources to execute and interact using safe, standard APIs and event mechanisms. Enable Lightning Locker for AppExchange packages that contain Lightning components or applications.

### Insufficient Escaping in Lightning Components

Each component in a bundle is responsible for sanitizing the input provided to it by parent components, apps, or in URL parameters.

### Asynchronous Code in Components

Hackers can manipulate the timing of asynchronous code to produce malicious results. To preserve current execution context, wrap asynchronous function calls or batch actions into a single request.

## Loading JavaScript Files from Third-Party Endpoints

Avoid dynamically loading third-party JavaScript files from content delivery networks (CDNs). Instead, load the code from the static resources folder of your package.

Dynamically loading third-party JavaScript files from CDNs or other third parties isn't permitted for two reasons.

- You must version your entire solution with a package version ID so that there's a well-defined product to review and track. If your solution dynamically loads code from third-party endpoints, the externally managed code can change without the package version ID changing. The administrator and the Salesforce security review team aren't made aware of the change.

Salesforce can't ensure that the third-party code continues to safeguard against the latest security vulnerabilities. To ensure that the code is subject to package version control, dynamically load the code from the static resources folder of your package. You can't change packaged code without changing the package version ID. Plus, version ID changes signal to administrators and the AppExchange security team that the code changed.

- Dynamically loading code from a third-party endpoint grants that endpoint the ability to inject code into any Salesforce org in which the package is installed. Only dynamically load code from Salesforce approved CDNs, where Salesforce manages the code, rather than the partner.

At a high level, the solution is:

1. Save third-party JavaScript files in static resources.

2. Add the resources to your solution package.
3. Load each JavaScript file from a `$Resource` URL.

## Visualforce Example

These code snippets depict the security violation and how to fix it in Apex and for Lightning components in Aura. This Visualforce code isn't secure because jQuery is loaded from a third-party source.

```
<apex:includescript value="https://code.jquery.com/jquery-3.2.1.min.js"/>
```

This Visualforce code is secure because it loads a version of jQuery from the static resources folder of your package using a `$Resource` URL.

```
<apex:includeScript value="{! $Resource.jquery }"/>
```

## Aura Example

This Aura component code isn't secure because jQuery is directly loaded from a third-party source.

```
<aura:component>
    <ltng:require afterScriptsLoaded="{!c.initializeUI}"
        scripts="https://code.jquery.com/jquery-2.2.0.min.js"/>
</aura:component>
```

This Aura component code is secure because jQuery is loaded from the solution package and referenced as a static resource using a `$Resource` URL.

```
<aura:component>
    <ltng:require afterScriptsLoaded="{!c.initializeUI}"
        scripts="{!$Resource.jsLibraries + '/jsLibOne.js'}/>
</aura:component>
```

## Loading Third-Party CSS in Lightning Components

Include cascading style sheets (CSS) and other resources in static resources rather than loading from a third party.

This requirement is enforced for the same reasons outlined in [Loading JavaScript Files from Third-Party Endpoints](#). The entire solution must be under version control, and the org administrator and Salesforce security review team must be aware of the change.

Using the `<link>` tag to load an external CSS resource violates this security policy.

At a high level, the solution is:

1. Save third-party CSS files in static resources.
2. Add the resources to your solution package.
3. Reference the CSS using a `<ltng:require>` tag in your .cmp or .app markup.

For more information, see [Using External CSS](#) in the Lightning Aura Components Developer Guide.

## Aura Example

These code snippets depict the security violation and how to fix it in a Lightning component in Aura. This Aura component code isn't secure because it uses the `<link>` tag to load an external CSS resource.

```
<aura:component>
    <link rel="stylesheet" href="https://example.com/styles.css" type="text/css">
</aura:component>
```

This Aura component code uses `<ltng:require>`, which is a more secure way to reference an external CSS resource that you uploaded as a static resource.

```
<aura:component>
    <ltng:require styles="{!$Resource.SLDSv1 +
'/assets/styles/lightning-design-system-ltng.css'}" />
</aura:component>
```

## Using CSS Outside Components

The Salesforce Platform tries to ensure that each namespace is an isolated sandbox, but isolation can't always be guaranteed. Where a namespace isolation breach occurs, one component can steal clicks from another component, or otherwise interfere with another component's intended use. To prevent this type of abuse, don't use CSS directives known to be incompatible with style isolation in your components.

## CSS Example

This CSS code is vulnerable because it uses absolute positioning, which is incompatible with style isolation.

```
#some_element {
    position: absolute;
    right: 80px;
    top: 160px;
}
```

This CSS code prevents the vulnerability by using relative positioning.

```
#some_element_revised {
    position: relative;
    right: 80px;
    top: 160px;
}
```

For more information, see [Tips for CSS in Components](#) in the Lightning Aura Components Developer Guide.

## Running JavaScript in the Salesforce Domain

JavaScript code from multiple vendors can run in the same origin. To prevent code interference, vendor JavaScript code is sandboxed. Don't attempt to break out of a sandbox or run code outside your origin. Use Visualforce, Aura, or Lightning Web Components, which run in the proper origin.

Many different types of JavaScript code run in a Salesforce org, including unpackaged customer code, Salesforce code, and packaged code. Typically, the code is from multiple vendors that have no way of collaborating with each other. If their code runs in the same origin, code from one vendor can interfere with other vendors' code.

To prevent code interference, vendor JavaScript code is sandboxed. With Visualforce solutions, JavaScript code is sandboxed in unique, vendor-specific origins. With Lightning solutions and Lightning Web Components (LWCs), JavaScript is sandboxed in unique, vendor-specific lockers.

Any attempt to break out of a sandbox and run code outside your origin is a secure coding violation. A secure coding violation includes attempts to run vendor-written JavaScript code in the Salesforce origin via homepage components, web links, or custom buttons.

In most situations, you can achieve the same functionality by using Visualforce, Aura, or Lightning Web Components, which run in the proper origin.

## Metadata Example

The metadata in this example represents a custom object. A web link within this custom object is defined using the `REQUIRESSCRIPT` statement. In a managed package, using `REQUIRESSCRIPT` is a security vulnerability because the vendor is injecting its code into a Salesforce origin. Managed packages must stay within their namespace sandbox and can't execute scripts outside this sandbox.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
  <actionOverrides>
    <actionName>Accept</actionName>
    <type>Default</type>
  </actionOverrides>
  <webLinks>
    <fullName>Add_to_List</fullName>
    <openType>onClickJavaScript</openType>
    <url>{!REQUIRESSCRIPT(&quot;/soap/ajax/30.0/connection.js&quot;)}</url>
  </webLinks>
  ...
</CustomObject>
```

Instead of embedding the code directly in the object, create a Visualforce button in a Visualforce Aura component, or use a Lightning Web Component.

## Exposing Secret Data When Debugging

In production environments, logging secret data with debug statements is a security vulnerability. Don't log secret data, sensitive information, passwords, keys, or stack traces in production environments. Redact the data or omit it from the logs.

Revealing secret data with debug statements makes it difficult for the Salesforce org admin to control access to the data. Typically, the profiles permitted to view logs aren't the same profiles that are permitted to view secrets.

## Apex Example

In this Apex code, `authenticationToken` is a cryptographic secret written to the debug log. To avoid this vulnerability, remove the `system.debug` statement from the production code.

```
if (varCount > 0){
    sensitiveUserData = JSON.serialize(AssignUsrs);
    ReqSignature = RequestWrapper.generateHmacSHA256Signature(sensitiveUserData,
authenticationToken);
    system.debug('Token--->' + authenticationToken);
}
```

## Storing Sensitive Data Insecurely

Follow enterprise security standards when you export data from the Salesforce Platform and when you store secret data in the platform.

Insecure sensitive data storage provides many avenues for hackers to pose threats. For example, an org administrator is the only person who is supposed to know the API key. Hackers can use an exposed key to communicate data over admin channels to remote endpoints.

Salesforce takes threats to data that originate in your solution seriously. A data breach or loss caused by a vulnerability in your solution jeopardizes your relationship with Salesforce.

Follow the enterprise standards in [Storing Sensitive Data](#) when:

- Exporting customer data from the Salesforce platform.
- Storing secrets such as cryptographic keys, session ids, or passwords in the Salesforce Platform.

## Metadata Example

The metadata in this example represents a custom object. This custom object definition isn't secure because the `<visibility>` tag for the API key field is set to `Public`. The field can be viewed in plain text.

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
  <fields>
    <fullName>apiKey__c</fullName>
    <externalId>false</externalId>
    <fieldManageability>DeveloperControlled</fieldManageability>
    <label>apiKey</label>
    <length>50</length>
    <required>false</required>
    <type>Text</type>
    <unique>false</unique>
  </fields>
  <label>Phone Verify Setting</label>
  <pluralLabel>Phone Verify Settings</pluralLabel>
  <visibility>Public</visibility>
</CustomObject>
```

When storing a secret in a custom object, such as an API key, encrypt it. Store the encryption key separately in a protected custom setting or a protected custom metadata API field.

## Using Software That Has Known Vulnerabilities

Using software that has documented common vulnerabilities and exposures (CVE) related to your use cases is a security vulnerability. If your solution has known vulnerabilities, test and deploy security patches as soon as they're available. If your solution uses software that has CVE-documented vulnerabilities unrelated to your use cases, prepare false positive documentation.

Hackers are quick to attack disclosed software vulnerabilities. Most vendors provide patches or updates for vulnerabilities discovered in their software. To find out if your solution uses software with known vulnerabilities, check the [Common Vulnerabilities and Exposures \(CVE\) database](#).

Apply all patches or updates related to your solution's use cases. If the vulnerabilities are unrelated to your use cases, document them as false positives. Explain why it's safe for your solution to use the vulnerable software. Our security review team uses this information when deciding whether to approve the software for use in your solution. Learn more in [False Positives](#).

## Using Sample Code in Production

Only use sample code as an educational tool in preparation for developing your own application. When building your production code, always write the code yourself. Avoid copying code from sources that you don't directly control.

There's great sample code available to developers all over the internet. While useful in learning best practices or new technologies, don't directly include sample code in production packages. Direct reuse can propagate vulnerabilities throughout many packages, whether intentional or not on the part of the sample code author.

## Bypassing Object-Level and Field-Level Access Settings

Design your solutions to enforce the org's create, read, update, and delete (CRUD) and field-level security (FLS) settings on standard and custom objects.

On the Salesforce Platform, you can configure CRUD access and FLS on profiles and permission sets. CRUD settings determine which objects a user can access. FLS determines which object fields a user can access. Use CRUD and FLS to restrict access to standard and custom objects and individual fields.

Customers expect that your solution doesn't violate the settings they have set in their orgs. Design your solutions to enforce the org's CRUD and FLS settings on standard and custom objects. Also, ensure that your solution gracefully handles situations where a user's access is restricted.

In certain use cases, it's acceptable to bypass CRUD and FLS, such as when:

- Creating Roll-Up summaries or aggregates that don't directly expose the data.
- Modifying custom objects or fields, such as logs or system metadata, so that they aren't directly accessible to the user via CRUD or FLS.
- Accessing objects from a high-privileged method, a method that non-admin users can't access.
- Denying guest user access to underlying objects when your solution is a community or site.
- Accessing custom objects belonging to your namespace with a bespoke security policy. In this case, document the policy as part of your AppExchange security review submission.

To learn more about CRUD and FLS enforcement, check out [Secure Server-Side Development module](#) on Trailhead.

## Apex Example

In this Apex code, the `insert account` data manipulation language (DML) statement runs without checking if the user has create access permission for the Account object. The code doesn't enforce the org's access settings.

```
public static Account createIndividualModalData(String name, String email, String mobile)
{
    RecordType recordType = [Select Id from RecordType where DeveloperName =
'IndustriesIndividual' and SubjectType = Account'];
    Account account = new Account();
    account.Name = name;

    if(recordType != null) account.RecordTypeId = recordType.id;
    insert account;
    ...
}
```

This Apex code is more secure because it enforces the org's access settings. It calls the `isCreatable()` method before the `insert account` DML statement executes. If `isCreatable()` returns true, the user has create access permission for the Account object and the `insert account` statement executes. Otherwise, an insufficient-access error is reported.

```
public static Account createIndividualModalData(String name, String email, String mobile)
{
    RecordType recordType = [Select Id from RecordType where DeveloperName =
'IndustriesIndividual' and ObjectType = 'Account'];
    Account account = new Account();
    account.Name = name;

    if(recordType != null) account.RecordTypeId = recordType.id;

    if (Schema.sObjectType.Account.isCreatable()) {
        insert account;
    } else {
        ApexPages.addMessage(new ApexPages.Message(ApexPages.Severity.ERROR, 'Error:
Insufficient Access'));
    }
    ...
}
```

## Bypassing Sharing Rules in Apex

Respect profile-based permissions, field-level security, sharing rules, and org-wide defaults in your Apex code.

The Salesforce Platform makes extensive use of data-sharing rules. Each object can have unique permissions that indicate which users and profiles can read, create, edit, and delete records of that object type. These restrictions are enforced when your code uses a standard controller.

However, a custom Apex class or Visualforce page doesn't intrinsically respect built-in profile permissions, field-level security restrictions, or sharing rules. By default, an Apex class can read and update all data within an org.

In your Apex code, don't expose sensitive data that is otherwise hidden from users. Respect profile-based permissions, field-level security, sharing rules, and org-wide defaults.

Follow these general rules for correctly enforcing sharing.

- Declare with `sharing` on all global classes or classes containing `@NamespaceAccessible` methods. Don't omit a sharing declaration or use `without sharing` on these entrypoints to your solution.
- For controller classes that aren't global or marked `@NamespaceAccessible`, either declare the class as `with inherited sharing` or `with sharing`. Don't omit a sharing declaration or use `without sharing` on these entrypoints to your solution.
- Declare all classes that directly perform data access operations as `with sharing`. If no class in your solution is marked `without sharing`, then `with inherited sharing` can also be used.

However, there are some notable exceptions. Don't follow the general rules when:

- You're building a site or community and want to deny guest-user access to data.
- You're accessing custom objects belonging to your namespace with a bespoke security policy. In this case, document the policy as part of your AppExchange security review submission documents. This exception doesn't apply to standard objects. The org admin solely owns security policy for standard objects.

## Apex Example

In this Apex code, the `with sharing` keyword isn't added to class header. By default, sharing rules aren't enforced.

```
public class maincontroller {
    @AuraEnabled public static String saveJobApplication(String vacId, String userId) {
        ...
    }
}
```

In this Apex code, the `with sharing` keyword is used. Sharing rules are enforced.

```
public with sharing class maincontroller {
    @AuraEnabled public static String saveJobApplication(String vacId, String userId) {
        ...
    }
}
```

To learn more about sharing rules enforcement, check out the [Secure Server-Side Development module](#) on Trailhead.

## SOQL Injection Due to Insecure Database Query Construction

To prevent Salesforce Object Query Language (SOQL) injection, use bind variables and input sanitation.

SOQL injection is a vulnerability in which a user directly controls portions of a SOQL database query. SOQL queries executed in Apex don't respect user permissions. Therefore, SOQL injections can be used to elevate users' privileges and allow them to access to data beyond their user permissions.

The two types of SOQL injection vulnerabilities require different protection approaches.

In the first type, the user supplies an incorrect table or field name to query against. When user data identifies a field or table name, you must verify that the user has permission to access the named table or field. Keep in mind that this type isn't a quoted context.

In the second type, the user supplies a portion of a quoted WHERE clause. When user data is inserted into a quoted string context, the data can break out of the quoted context. The preferred protection approach is to use bind variables. Alternatively, you can use `EscapeSingleQuotes()`. Both of these approaches prevent the user data from breaking out of the quoted context.

Never allow users to supply portions of SOQL queries other than field names, table names, and WHERE clause inputs.

Avoid executing user-generated queries in Apex, where they run in system mode. If you must generate more complex client-side SOQL, use the REST or SOAP API, which make SOQL calls safely.

To learn more about SOQL injection and how to prevent it in your code, check out the [Secure Server-Side Development module](#) on Trailhead.

## SOQL Example

This SOQL statement is insecure because it's built by concatenating a string with embedded user input. No input sanitization occurs before the `database.query` statement executes.

```
string objType = id.valueOf(deduped[0].recordId).getObjectType().getDescribe().getName();
string soql = 'select id, ' + string.join(fields, ', ') + ' from ' + objType + ' where
id in: lrlIDs';
list<subject> records = database.query(soql);
```

If you must use dynamic SOQL, use the `EscapeSingleQuotes()` method to sanitize user-supplied input. This method adds the escape character (`\`) to all single quotation marks in the user-supplied string. Adding the escape character ensures that all single quotation marks are treated as enclosing strings instead of as database commands.

```
string objType =
escapeSingleQuotes(id.valueOf(deduped[0].recordId).getSObjectType().getDescribe().getName());
string soql = 'select id, ' + string.join(fields, ', ') + ' from ' + objType + ' where
id in: lrlIDs';
list<object> records = database.query(soql);
```

## Cross-Site Request Forgery

A cross-site request forgery (CSRF) is an attack that forces an end user to execute unwanted actions during their authenticated web application session. To protect against CSRF, use `confirmationTokenRequired`, or trigger state changes with user actions.

All form requests made on the Salesforce Platform are protected. Insert, delete, update, and upsert state change operations triggered by user action, such as a button click, are also protected.

However, state change or data manipulation language (DML) operations triggered on page instantiation execute before the rest of the page loads, and they bypass the platform's default CSRF protection. State change and DML operations in class constructors are vulnerable if they're triggered from:

- Visualforce pages
- Lightning web components (LWC)
- Aura
- Any methods called from the action parameter of a Visualforce page

## Apex Example

This Visualforce page is vulnerable to CSRF because the `!init` action is triggered on page initialization.

```
<apex:page controller="maincontroller" action="{!init}">

public pageReference init(){

    UserSetting__c accountToUpdate;
    pageReference p = page.mainview;
    // Retrieve the password and redirect query string parameters from the current page URL

    String password = ApexPages.currentPage().getParameters().get('password');
    String redirect = ApexPages.currentPage().getParameters().get('redirect');
    if(string.isBlank(redirect)){
        p.getParameters().put('redirect', '/home/home.jsp');
        p.setRedirect(true);
    } else {
        p.getParameters().put('redirect', redirect);
    }
    if(string.isBlank(password)){
        p.getParameters().put('password', 'blank');
        p.setRedirect(true);
    } else {
        p.getParameters().put('password', password);
        accountToUpdate = [SELECT password__c FROM UserSetting__c LIMIT 1];
```

```

        accountToUpdate.password__C = password;
        update accountToUpdate;
    }
    if(p.getRedirect() == true){
        return p;
    }
    else {
        return null;
    }
}

```

A hacker can craft a URL containing parameters that alter database statements, allowing them to perform malicious actions of their choosing. When a user opens such a URL while logged in to your app, the code executes using the hacker's chosen URL parameters. The unintended database actions execute from the context of the victim's browser.

## Visualforce Page Protection

To protect against the CSRF vulnerability in a Visualforce page when state change or DML operations execute on page initialization, enable the `confirmationTokenRequired` boolean metadata field in the Visualforce page.

If `confirmationTokenRequired` is set to true, GET requests to the page require a CSRF token in the URL. If the token is omitted, the page is inaccessible.

The default setting is false, which removes Apex's built-in CSRF token protection. You can configure this field by going to relevant Visualforce page settings in org setup.

For more info about `confirmationTokenRequired`, refer to [ApexPage](#) in the Metadata API Developer Guide.

## Lightning and LWC CSRF Protection

Don't perform any state change or DML operations in an Apex controller during instantiation of Lightning or LWC. Instead, trigger a state change with a user action, such as a button click. To learn more about CSRF and how to prevent it in your code, check out the [Secure Server-Side Development module](#) on Trailhead.

## Open Redirects

An open redirect occurs when an application dynamically redirects to a user-controlled parameter value without any validation. Prevent open redirects by using hardcoded redirects.

Open redirects are also known as arbitrary or unvalidated redirects. This vulnerability is used in phishing attacks to redirect users to any URL.

## Apex Example

In this function definition, the `String.redirect` statement retrieves the `redirect` URL parameter for the current page. The parameter is used to craft a redirection URL, and then to perform a client-side redirect to the crafted URL.

```

public PageReference changepassword() {
    PageReference savePage;
    String redirect = ApexPages.currentPage().getParameters().get('redirect');
    redirect = (redirect == NULL) ? '/home/home.jsp' : redirect;
    savePage = new PageReference(redirect);
    savePage.setRedirect(true);
}

```

```
    return savePage;
}
```

The `<apex:form>` Visualforce markup view triggers the `changepassword` action, which results in an open redirect vulnerability in a package.

```
<apex:form>
  Redirection action: <apex:inputText value="{!userInput}" />
  <br/><apex:commandButton value="Submit" action="{!changepassword}" />
</apex:form>
```

## Revised Code

Open redirects expose your redirection parameters to potential attackers. You can prevent open redirects using multiple strategies. One strategy is to use hardcoded redirects. In a hardcoded redirect, you set the value explicitly as shown in this example:

```
public PageReference changepassword() {
    PageReference savePage;
    savePage = new PageReference('/home/home.jsp');
    savePage.setRedirect(true);
    return savePage;
}
```

To learn more about open redirects and how to prevent them in your code, check out the [Secure Server-Side Development module](#) on Trailhead.

## Lightning LockerService Disabled

Lightning LockerService is a critical security feature for Lightning code. It provides component isolation that allows code from many sources to execute and interact using safe, standard APIs and event mechanisms. Enable Lightning Locker for AppExchange packages that contain Lightning components or applications.

Lightning LockerService is enabled for all custom Lightning web components. The service was activated for customers in the Summer '17 release. Lightning LockerService isn't enforced for components that use API version 39.0 and lower, which covers any component created before Summer '17. When a component is set to at least API version 40.0, it's enabled. New AppExchange security reviews and periodic re-reviews require components to be version 40.0 or higher so that Locker is enabled.

## Metadata Example

In this component's `<AuraDefinitionBundle>` metadata, the `<apiVersion>` field sets the API version to 39.0. LockerService is disabled for components that use API version 39.0 and lower.

```
<?xml version="1.0" encoding="UTF-8"?>
<AuraDefinitionBundle xmlns="http://soap.sforce.com/2006/04/metadata">
  <apiVersion>39.0</apiVersion>
  <description>My Component</description>
</AuraDefinitionBundle>
```

In this component's revised `<AuraDefinitionBundle>` metadata, the `<apiVersion>` field sets the API version to 40.0. LockerService is enforced for components that use API version 40.0 and higher.

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

```
<apiVersion>40.0</apiVersion>
<description>My Component</description>
</AuraDefinitionBundle>
```

For more information, read the [Summer 2017 Release Notes](#) and [Security with Lightning Locker](#) in the Lightning Web Components Developer Guide.

## Insufficient Escaping in Lightning Components

Each component in a bundle is responsible for sanitizing the input provided to it by parent components, apps, or in URL parameters.

The security boundary of an individual component is the component bundle. Each component in a bundle is responsible for sanitizing the input provided to it by parent components, apps, or in URL parameters. Public or global component attributes are assumed to contain attacker-controlled inputs unless sanitized by the component in an `onInit` handler.

Failure to sanitize inputs can lead to cross-site scripting (XSS) or URL redirection attacks.

### Aura Example

In this Aura code, a component reads data from a global attribute and then renders it to the document object model (DOM) without sufficient escaping. One parameter has the tag `unescapedHTML`, which is open to exploitation. A hacker or malware can inject HTML or JavaScript into the view and trigger a cross-site scripting (XSS) attack.

```
<aura:component controller="name_NewsController" access="global" extends="c:name_Name"
implements="force:appHostable,flexipage:availableForAllPageTypes,forceCommunity:availableForAllPageTypes">

    <aura:handler name="baseReady" event="c:name_Name" action="{!c.doInit}"/>
    ...
    <aura:attribute name="newsDetails" type="String" default="" access="global"/>
    ...
    <div class="slds-col_padded slds-size_1-of-1 textDetail">
        <div class="slds-text-longform">
            <aura:unescapedHtml aura:Id="newsDetail" value="{!v.newsDetails}"/>
        </div>
    </div>
    ...
</aura:component>
```

This Aura component code is secure because it doesn't use the `unescapedHTML`.

```
<aura:component controller="name_NewsController" access="global" extends="c:name_Name"
implements="force:appHostable,flexipage:availableForAllPageTypes,forceCommunity:availableForAllPageTypes">

    <aura:handler name="baseReady" event="c:name_Name" action="{!c.doInit}"/>
    ...
    <aura:attribute name="newsDetails" type="String" default="" access="global"/>
    ...
    <div class="slds-col_padded slds-size_1-of-1 textDetail">
        <div class="slds-text-longform">
            <aura:Id="newsDetail" value="{!v.newsDetails}"/>
        </div>
    </div>
    ...
</aura:component>
```


For more info, refer to [Lightning Security](#) in the Secure Coding Guide.

## Asynchronous Code in Components

Hackers can manipulate the timing of asynchronous code to produce malicious results. To preserve current execution context, wrap asynchronous function calls or batch actions into a single request.

When you use an asynchronous function such as `setTimeout()` and `setInterval()` to reference a component, you exit the framework's lifecycle. If you navigate elsewhere in the user interface while asynchronous code is executing, the framework unrenders and destroys the component that made the asynchronous request. You can still have a reference to that component, but it's no longer valid. Hackers exploit this vulnerability in harmful ways, for example, crash an app.

To reenter the framework safely, wrap the code in the `$A.getCallback()` function. Then, to ensure that the component is still valid, use the `component.isValid()` function before executing anything in the callback. Alternatively, batch multiple actions into one request by using `enqueueAction()`.

 **Note:** This vulnerability doesn't apply to components created against the Summer '17 release (API v40.0) or later.

These examples depict the security violation and how to fix it.

### Aura Example

The `setInterval()` function gives you access to the document object model (DOM). However, accessing the DOM with `setInterval()` occurs in a context outside of the Lightning framework. There are no guarantees about the parent component's state—it's possible the function doesn't have a parent component at all. If the state changes, the callback function can act on data that it doesn't own, or it can wait for data that never shows up. In these scenarios, the app throws an error message that halts the entire Salesforce page, and the component stops responding.

```
vars.Timer = setInterval(function(){ helper.action(component); },1);
```

### Revised Code Using getCallback() Example

To reenter the framework safely, wrap the code in the `$A.getCallback()` function. Then, to ensure that the component is still valid, use the `component.isValid()` function before executing anything in the callback.

Use `$A.getCallback()` to wrap any code that accesses a component outside the normal re rendering lifecycle, such as in a `setTimeout()` or `setInterval()` call. `$A.getCallback()` preserves the current execution context and grants the correct access level to the asynchronous code. Otherwise, the framework loses context and only allows access to global resources.

```
window.setTimeout(  
    $A.getCallback(function() {  
        if(cmp.isValid()){  
            cmp.set("v.visible", true);  
        }  
    }  
), 5000  
);
```

## Revised Code Using enqueueAction() Example

Alternatively, use `enqueueAction()`, which adds the server-side controller action to the queue of actions to be executed. Rather than sending a separate request for each individual action, the framework processes the event chain and batches the actions in the queue into one request. The actions are asynchronous and have callbacks.

```
var action = component.get("c.usually_a_server_side_controller");
action.setCallback(this, function() {response} {...});
$A.enqueueAction(action2);
```

To learn more, check out our [Secure Client-Side Development module](#) on Trailhead.

## 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



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

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

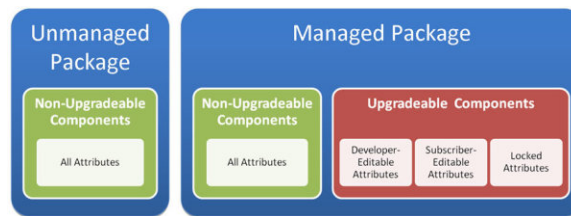
Managed packages also offer the following benefits:

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

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	<p>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:</p> <ul style="list-style-type: none"> <li>It has not been uploaded.</li> <li>It has been uploaded with <code>Managed - Beta</code> 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.</li> </ul> <p> <b>Note:</b> 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.</p>

State	Description
 Managed - Released	The package or component was created in the current Salesforce organization and is managed. It is also uploaded with the <b>Managed - Released</b> option selected, indicating that it can be published on AppExchange and is publicly available. Note that once you have moved a package to this state, some properties of the components are no longer editable for both the developer and installer.  This type of release is considered a major release.
Patch	If you need to provide a minor upgrade to a managed package, consider creating a patch instead of a new major release. A patch enables a developer to change the functionality of existing components in a managed package. Subscribers experience no visible changes to the package.  This type of release is considered a patch release.
 Managed - Installed	The package or component was installed from another Salesforce organization but is managed.

A developer can refine the functionality in a managed package over time, uploading and releasing new versions as the requirements evolve. This might involve redesigning some of the components in the managed package. Developers can delete some, but not all, types of components in a Managed - Released package when upgrading it. For details, see [Delete Components in Managed Packages](#) on page 359.

## 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 78. 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](#).

### 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

 **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**.

## 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 the Salesforce Extensions for Visual Studio Code 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 38. 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.

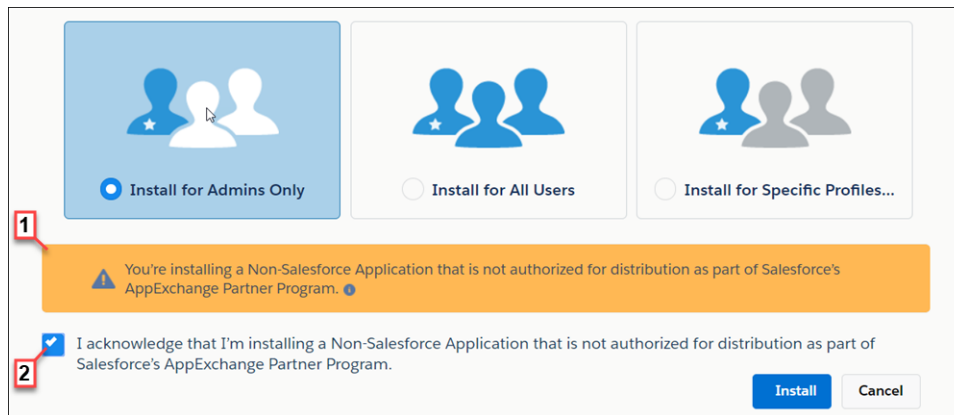
 **Note:** Some related components are automatically included in the package even though they might not display in the Package Components list. For example, when you add a custom object to a package, its custom fields, page layouts, and relationships with standard objects are automatically included. For a complete list of components, see [Components Automatically Added to Packages](#) on page 55.

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

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

The following components are available in a managed package.

**Upgradeable**

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

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

**Subscriber Deletable**

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

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

**Developer Deletable**

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

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

**Protectable**

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

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

**IP Protection**

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

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

Component	Upgradeable	Subscriber Deletable	Developer Deletable	Protectable	IP Protection
<b>Action</b>	Yes	No	No	No	No
<b>Analytics Application</b>	No	No	No	No	No
<b>Analytics Dashboard</b>	No	No	No	No	No
<b>Analytics Dataflow</b>	Yes	No	No	No	No
<b>Analytics Dataset</b>	No	No	No	No	No
<b>Analytics Dataset Metadata</b>	No	No	No	No	No
<b>Analytics Lens</b>	No	No	No	No	No

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

<sup>1</sup> Requires a Service Cloud license.

Component	Upgradeable	Subscriber Deletable	Developer Deletable	Protectable	IP Protection
<b>Custom Permission</b>	Yes	No	No	Yes	No
<b>Custom Report Type</b>	Yes	No	No	No	No
<b>Custom Setting</b>	Yes	Yes <sup>2</sup>	Yes <sup>3</sup>	Yes	Yes
<b>Dashboard</b>	No	Yes	Yes	No	No
<b>Data Classification on Custom Fields</b>	No	Yes	Yes	No	No
<b>Document<sup>7</sup></b> (10-MB limit)	No	Yes	Yes	No	No
<b>Experience Builder Template</b>	Yes	No	Yes	No	No
<b>Experience Builder Theme</b>	Yes	No	Yes	No	No
<b>External Data Source</b>	Yes	No	No	No	No
<b>Email Template (Classic)</b>	No	Yes	Yes	No	No
<b>Email Template (Lightning)</b>	No	No	Yes	No	No
<b>Email Template (Lightning):</b> created in Email Template Builder	FlexiPage only	No	No	No	No
<b>External Services<sup>5</sup></b>	Yes	No	Yes	No	No
<b>Field Set</b>	Yes	Yes <sup>2</sup>	Yes <sup>3</sup>	No	No
<b>Flow</b>	Yes	No	No	No	Yes, except templates
<b>Folder</b>	No	Yes	Yes	No	No
<b>Global Picklist</b>	Yes	Yes	Yes	No	No
<b>Home Page Component</b>	Yes	No	No	No	No
<b>Home Page Layout</b>	No	Yes	Yes	No	No
<b>Inbound Network Connection</b>	Yes	Yes <sup>9</sup>	Yes <sup>9</sup>	No	No

Component	Upgradeable	Subscriber Deletable	Developer Deletable	Protectable	IP Protection
Letterhead	No	Yes	Yes	No	No
Lightning Application	Yes	Yes <sup>8</sup>	Yes <sup>3</sup>	No	No
Lightning Bolt	Yes	No	Yes	No	No
Lightning Component (Aura and Lightning Web Component)	Yes	Yes <sup>8</sup>	Yes <sup>3</sup>	No	No
Lightning Event	Yes	No	No	No	No
Lightning Interface	Yes	No	No	No	No
Lightning page	Yes	No	No	No	No
List View	No	Yes	Yes	No	No
Named Credential <sup>5</sup>	Yes	No	No	No	No
Next Best Action Recommendation Strategy	Yes	No	No	No	Yes, except templates
Outbound Network Connection	Yes	Yes <sup>9</sup>	Yes <sup>9</sup>	No	No
Page Layout	No	Yes	Yes	No	No
Path Assistant	Yes	Yes	Yes	No	No
Permission Set	Yes	Yes <sup>2</sup>	Yes <sup>3</sup>	No	No
Platform Cache	No	No	No	No	No
Platform Event Channel	No	No	No	No	No
Platform Event Channel Member	No	No	No	No	No
Process	See Flow				
Prompts (In-App Guidance)	Yes	No	No	No	No
Record Type	Yes	Yes <sup>2</sup>	Yes <sup>3</sup>	No	No
Remote Site Setting	No	Yes	Yes	No	No
Report	No	Yes	Yes	No	No

Component	Upgradeable	Subscriber Deletable	Developer Deletable	Protectable	IP Protection
<b>Reporting Snapshot</b>	No	Yes	Yes	No	No
<b>Salesforce IoT</b>	Yes	No	No	No	No
<b>S-Control<sup>7</sup></b> (10-MB limit)	Yes	Yes <sup>2</sup>	Yes <sup>3</sup>	No	No
<b>Static Resource</b>	Yes	Yes <sup>2</sup>	Yes <sup>3</sup>	No	No
<b>Embedded Service Deployment<sup>62</sup></b>	No	No	No	No	No
<b>Tab</b>	Yes	Yes <sup>2</sup>	Yes <sup>3</sup>	No	No
<b>TimeSheetTemplate</b>	Yes	Yes <sup>2</sup>	Yes <sup>3</sup>	No	No
<b>Translation</b>	Yes	No	No	No	No
<b>Validation Rule</b>	Yes	Yes <sup>2</sup>	Yes <sup>2</sup>	No	No
<b>Visualforce Component</b>	Yes	Yes <sup>4</sup>	Yes <sup>3</sup>	No	Yes
<b>Visualforce Page</b>	Yes	Yes <sup>2</sup>	Yes <sup>3</sup>	No	No
<b>Workflow Field Update</b>	Yes	No	Yes, if protected	Yes	No
<b>Workflow Outbound Message</b>	Yes	No	Yes, if protected	Yes	No
<b>Workflow Rule</b>	Yes	No	No	No	No
<b>Workflow Task</b>	Yes	No	Yes, if protected	Yes	No

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

<sup>2</sup> 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.

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

<sup>4</sup> 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.

---

<sup>2</sup> Requires a Service Cloud license.

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

<sup>6</sup> 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.

<sup>7</sup> The combined size of S-Controls and documents must be less than 10 MB.

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

<sup>9</sup> You can only delete connections that are in an unprovisioned state.

### Component Attributes and Behaviors

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

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

### Deleting Visualforce Pages and Global Visualforce Components

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

### Deleting Lightning Components

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

SEE ALSO:

[Editing Components and Attributes After Installation](#)

[Components Automatically Added to Packages](#)

[Delete Components from First-Generation Managed Packages](#)

## Components Available in Unmanaged Packages

Not all components can be packaged for distribution. The following table lists:

- Components that are available in an unmanaged package
- How the component is included in the package
- 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
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 (Classic)	Explicitly In a folder: Implicitly	Yes

Component	Packaged Explicitly or Implicitly	Automatic Renaming
<b>External Data Source</b>	Explicitly Referenced by an external object: Implicitly Assigned by a permission set: Implicitly	No
<b>Flow Definition</b>	Implicitly	No
<b>Folder</b>	Explicitly	Yes
<b>Home Page Component</b>	Explicitly	No
<b>Home Page Layout</b>	Explicitly	No
<b>Inbound Network Connection</b>	Explicitly	No
<b>Letterhead</b>	Explicitly	Yes
<b>Lightning Application</b>	Explicitly	No
<b>Lightning Component</b>	Explicitly	No
<b>Lightning Event</b>	Explicitly	No
<b>Lightning Interface</b>	Explicitly	No
<b>Lightning Page</b>	Explicitly	No
<b>List View</b>	On a standard object: Explicitly On a custom object: Implicitly	Yes
<b>Named Credential</b>	Explicitly	No
<b>Outbound Network Connection</b>	Explicitly	No
<b>Page Layout</b>	On a standard object: Explicitly On a custom object: Implicitly	No
<b>Record Type</b>	On a standard object: Explicitly On a custom object: Implicitly	No
<b>Report</b>	Explicitly In a folder: Implicitly	Yes
<b>S-Control*</b> (10 MB limit)	Explicitly	No
<b>Static Resource</b>	Explicitly	No
<b>Tab</b>	Explicitly	No
<b>Translation</b>	Explicitly	No

Component	Packaged Explicitly or Implicitly	Automatic Renaming
<b>Validation Rule</b>	On a standard object: Explicitly On a custom object: Implicitly	No
<b>Visualforce Component</b>	Explicitly	No
<b>Visualforce Page</b>	Explicitly	No
<b>Workflow Email Alert</b>	Explicitly	No
<b>Workflow Field Update</b>	Explicitly	No
<b>Workflow Outbound Message</b>	Explicitly	No
<b>Workflow Rule</b>	Explicitly	No
<b>Workflow Task</b>	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
<b>Action</b>		<ul style="list-style-type: none"> <li>Action layout</li> <li>Predefined values for action fields</li> </ul>	<ul style="list-style-type: none"> <li>All fields</li> </ul>
<b>Apex Class</b>	<ul style="list-style-type: none"> <li>API Version</li> <li>Code</li> </ul>		<ul style="list-style-type: none"> <li>Name</li> </ul>
<b>Apex Sharing Reason</b>	<ul style="list-style-type: none"> <li>Reason Label</li> </ul>		<ul style="list-style-type: none"> <li>Reason Name</li> </ul>
<b>Apex Sharing Recalculation</b>		<ul style="list-style-type: none"> <li>Apex Class</li> </ul>	

Component	Developer Editable	Subscriber and Developer Editable	Locked
<b>Apex Trigger</b>	<ul style="list-style-type: none"> <li>API Version</li> <li>Code</li> </ul>		<ul style="list-style-type: none"> <li>Name</li> </ul>
<b>Application</b>	<ul style="list-style-type: none"> <li>Show in Lightning Experience (SalesforceClassic only)</li> <li>Selected Items (Lightning Experience only)</li> <li>Utility Bar (Lightning Experience only)</li> </ul>	<ul style="list-style-type: none"> <li>All attributes, except App Name and Show in Lightning Experience (Salesforce Classic only)</li> <li>All attributes, except Developer Name, Selected Items, and Utility Bar (Lightning Experience only)</li> </ul>	<ul style="list-style-type: none"> <li>App Name (SalesforceClassic only)</li> <li>Developer Name (Lightning Experience only)</li> </ul>
<b>Article Types</b>	<ul style="list-style-type: none"> <li>Description</li> <li>Label</li> <li>Plural Label</li> <li>Starts with a Vowel Sound</li> </ul>	<ul style="list-style-type: none"> <li>Available for Customer Portal</li> <li>Channel Displays</li> <li>Default Sharing Model</li> <li>Development Status</li> <li>Enable Divisions</li> <li>Grant Access Using Hierarchy</li> <li>Search Layouts</li> </ul>	<ul style="list-style-type: none"> <li>Name</li> </ul>
<b>Compact Layout</b>		<ul style="list-style-type: none"> <li>All attributes</li> </ul>	
<b>Connected App</b>	<ul style="list-style-type: none"> <li>Access Method</li> <li>Canvas App URL</li> <li>Callback URL</li> <li>Connected App Name</li> <li>Contact Email</li> <li>Contact Phone</li> <li>Description</li> <li>Icon URL</li> <li>Info URL</li> <li>Trusted IP Range</li> <li>Locations</li> <li>Logo Image URL</li> <li>OAuth Scopes</li> </ul>	<ul style="list-style-type: none"> <li>ACS URL</li> <li>Entity ID</li> <li>IP Relaxation</li> <li>Mobile Start URL</li> <li>Permitted Users</li> <li>Refresh Token Policy</li> <li>SAML Attributes</li> <li>Service Provider Certificate</li> <li>Start URL</li> <li>Subject Type</li> </ul>	<ul style="list-style-type: none"> <li>API Name</li> <li>Created Date/By</li> <li>Consumer Key</li> <li>Consumer Secret</li> <li>Installed By</li> <li>Installed Date</li> <li>Last Modified Date/By</li> <li>Version</li> </ul>
<b>Custom Button or Link</b>	<ul style="list-style-type: none"> <li>Behavior</li> </ul>	<ul style="list-style-type: none"> <li>Height</li> </ul>	<ul style="list-style-type: none"> <li>Display Type</li> </ul>

Component	Developer Editable	Subscriber and Developer Editable	Locked
	<ul style="list-style-type: none"> <li>• Button or Link URL</li> <li>• Content Source</li> <li>• Description</li> <li>• Display Checkboxes</li> <li>• Label</li> <li>• Link Encoding</li> </ul>	<ul style="list-style-type: none"> <li>• Resizeable</li> <li>• Show Address Bar</li> <li>• Show Menu Bar</li> <li>• Show Scrollbars</li> <li>• Show Status Bar</li> <li>• Show Toolbars</li> <li>• Width</li> <li>• Window Position</li> </ul>	<ul style="list-style-type: none"> <li>• Name</li> </ul>
<b>Custom Field</b>	<ul style="list-style-type: none"> <li>• Auto-Number Display Format</li> <li>• Decimal Places</li> <li>• Description</li> <li>• Default Value</li> <li>• Field Label</li> <li>• Formula</li> <li>• Length</li> <li>• Lookup Filter</li> <li>• Related List Label</li> <li>• Required</li> <li>• Roll-Up Summary Filter Criteria</li> </ul>	<ul style="list-style-type: none"> <li>• Chatter Feed Tracking</li> <li>• Help Text</li> <li>• Mask Type</li> <li>• Mask Character</li> <li>• Sharing Setting</li> <li>• Sort Picklist Values</li> <li>• Track Field History</li> </ul>	<ul style="list-style-type: none"> <li>• Child Relationship Name</li> <li>• Data Type</li> <li>• External ID</li> <li>• Field Name</li> <li>• Roll-Up Summary Field</li> <li>• Roll-Up Summary Object</li> <li>• Roll-Up Summary Type</li> <li>• Unique</li> </ul>
<b>Custom Label</b>	<ul style="list-style-type: none"> <li>• Category</li> <li>• Short Description</li> <li>• Value</li> </ul>		<ul style="list-style-type: none"> <li>• Name</li> </ul>
<b>Custom Object</b>	<ul style="list-style-type: none"> <li>• Description</li> <li>• Label</li> <li>• Plural Label</li> <li>• Record Name</li> <li>• Starts with a Vowel Sound</li> </ul>	<ul style="list-style-type: none"> <li>• Allow Activities</li> <li>• Allow Reports</li> <li>• Available for Customer Portal</li> <li>• Context-Sensitive Help Setting</li> <li>• Default Sharing Model</li> <li>• Development Status</li> <li>• Enable Divisions</li> <li>• Enhanced Lookup</li> <li>• Grant Access Using Hierarchy</li> </ul>	<ul style="list-style-type: none"> <li>• Object Name</li> <li>• Record Name Data Type</li> <li>• Record Name Display Format</li> </ul>

Component	Developer Editable	Subscriber and Developer Editable	Locked
		<ul style="list-style-type: none"> <li>Search Layouts</li> <li>Track Field History</li> </ul>	
Custom Permission	<ul style="list-style-type: none"> <li>Connected App</li> <li>Description</li> <li>Label</li> <li>Name</li> </ul>		
Custom Report Type	<ul style="list-style-type: none"> <li>All attributes except Development Status and Report Type Name</li> </ul>	<ul style="list-style-type: none"> <li>Development Status</li> </ul>	<ul style="list-style-type: none"> <li>Report Type Name</li> </ul>
Custom Setting	<ul style="list-style-type: none"> <li>Description</li> <li>Label</li> </ul>		<ul style="list-style-type: none"> <li>Object Name</li> <li>Setting Type</li> <li>Visibility</li> </ul>
Dashboard		<ul style="list-style-type: none"> <li>All attributes except Dashboard Unique Name</li> </ul>	<ul style="list-style-type: none"> <li>Dashboard Unique Name</li> </ul>
Document		<ul style="list-style-type: none"> <li>All attributes except Document Unique Name</li> </ul>	<ul style="list-style-type: none"> <li>Document Unique Name</li> </ul>
Email Template (Classic)		<ul style="list-style-type: none"> <li>All attributes except Email Template Name</li> </ul>	<ul style="list-style-type: none"> <li>Email Template Name</li> </ul>
Email Template (Lightning)			All attributes
Email Template (Lightning) created in Email Template Builder			All attributes
External Data Source	<ul style="list-style-type: none"> <li>Type</li> </ul>	<ul style="list-style-type: none"> <li>Auth Provider</li> <li>Certificate</li> <li>Custom Configuration</li> <li>Endpoint</li> <li>Identity Type</li> <li>OAuth Scope</li> <li>Password</li> <li>Protocol</li> <li>Username</li> </ul>	<ul style="list-style-type: none"> <li>Name</li> </ul>

Component	Developer Editable	Subscriber and Developer Editable	Locked
<b>Field Set</b>	<ul style="list-style-type: none"> <li>Description</li> <li>Label</li> <li>Available fields</li> </ul>	<ul style="list-style-type: none"> <li>Selected fields (only subscriber controlled)</li> </ul>	<ul style="list-style-type: none"> <li>Name</li> </ul>
<b>Flow</b>	<ul style="list-style-type: none"> <li>Entire flow</li> </ul>	<ul style="list-style-type: none"> <li>Flow Label</li> <li>Description</li> <li>Status</li> </ul>	<ul style="list-style-type: none"> <li>Flow API Name</li> <li>URL</li> </ul>
<b>Folder</b>		<ul style="list-style-type: none"> <li>All attributes except Folder Unique Name</li> </ul>	<ul style="list-style-type: none"> <li>Folder Unique Name</li> </ul>
<b>Home Page Component</b>	<ul style="list-style-type: none"> <li>Body</li> <li>Component Position</li> </ul>		<ul style="list-style-type: none"> <li>Name</li> <li>Type</li> </ul>
<b>Home Page Layout</b>		<ul style="list-style-type: none"> <li>All attributes except Layout Name</li> </ul>	<ul style="list-style-type: none"> <li>Layout Name</li> </ul>
<b>Inbound Network Connection</b>	<ul style="list-style-type: none"> <li>Connection Type</li> <li>Developer Name</li> <li>Description</li> <li>Master Label</li> <li>Region</li> </ul>	<ul style="list-style-type: none"> <li>Status</li> </ul>	
<b>Letterhead</b>		<ul style="list-style-type: none"> <li>All attributes except Letterhead Name</li> </ul>	<ul style="list-style-type: none"> <li>Letterhead Name</li> </ul>
<b>Lightning Application</b>	<ul style="list-style-type: none"> <li>API Version</li> <li>Description</li> <li>Label</li> <li>Markup</li> </ul>		Name
<b>Lightning Component</b>	<ul style="list-style-type: none"> <li>API Version</li> <li>Description</li> <li>Label</li> <li>Markup</li> </ul>		Name
<b>Lightning Event</b>	<ul style="list-style-type: none"> <li>API Version</li> <li>Description</li> <li>Label</li> </ul>		Name

Component	Developer Editable	Subscriber and Developer Editable	Locked
	<ul style="list-style-type: none"> <li>Markup</li> </ul>		
<b>Lightning Interface</b>	<ul style="list-style-type: none"> <li>API Version</li> <li>Description</li> <li>Label</li> <li>Markup</li> </ul>		Name
<b>Lightning Page</b>	<ul style="list-style-type: none"> <li>Lightning page</li> </ul>		
<b>Lightning Web Component</b>	<ul style="list-style-type: none"> <li>API Version</li> <li>Description</li> <li>Label</li> <li>Markup</li> </ul>		Name
<b>List View</b>		<ul style="list-style-type: none"> <li>All attributes except View Unique Name</li> </ul>	<ul style="list-style-type: none"> <li>View Unique Name</li> </ul>
<b>Named Credential</b>	<ul style="list-style-type: none"> <li>Endpoint</li> <li>Label</li> </ul>	<ul style="list-style-type: none"> <li>All attributes except Endpoint, Label, and Name</li> </ul>	<ul style="list-style-type: none"> <li>Name</li> </ul>
<b>Outbound Network Connection</b>	<ul style="list-style-type: none"> <li>Connection Type</li> <li>Developer Name</li> <li>Description</li> <li>Master Label</li> <li>Region</li> <li>Service Name</li> </ul>	<ul style="list-style-type: none"> <li>Status</li> </ul>	
<b>Page Layout</b>		<ul style="list-style-type: none"> <li>All attributes except Page Layout Name</li> </ul>	<ul style="list-style-type: none"> <li>Page Layout Name</li> </ul>
<b>Path Assistant</b>		IsActive field	SubjectType, SubjectProcessField, and RecordType
<b>Permission Set</b>	<ul style="list-style-type: none"> <li>Description</li> <li>Label</li> <li>Custom object permissions</li> <li>Custom field permissions</li> <li>Apex class access settings</li> <li>Visualforce page access settings</li> </ul>		<ul style="list-style-type: none"> <li>Name</li> </ul>



Component	Developer Editable	Subscriber and Developer Editable	Locked
<b>Platform Cache</b>	<ul style="list-style-type: none"> <li>Master Label</li> <li>Description</li> <li>Default Partition</li> </ul>	<ul style="list-style-type: none"> <li>Organization Capacity</li> <li>Trial Capacity</li> </ul>	<ul style="list-style-type: none"> <li>Developer Name</li> </ul>
<b>Record Type</b>	<ul style="list-style-type: none"> <li>Description</li> <li>Record Type Label</li> </ul>	<ul style="list-style-type: none"> <li>Active</li> <li>Business Process</li> </ul>	<ul style="list-style-type: none"> <li>Name</li> </ul>
<b>Remote Site Setting</b>		All attributes except Remote Site Name	<ul style="list-style-type: none"> <li>Remote Site Name</li> </ul>
<b>Report</b>		<ul style="list-style-type: none"> <li>All attributes except Report Unique Name</li> </ul>	<ul style="list-style-type: none"> <li>Report Unique Name</li> </ul>
<b>Reporting Snapshot</b>		<ul style="list-style-type: none"> <li>All attributes except Reporting Snapshot Unique Name</li> </ul>	<ul style="list-style-type: none"> <li>Reporting Snapshot Unique Name</li> </ul>
<b>S-Control</b>	<ul style="list-style-type: none"> <li>Content</li> <li>Description</li> <li>Encoding</li> <li>Filename</li> <li>Label</li> </ul>	<ul style="list-style-type: none"> <li>Prebuild in Page</li> </ul>	<ul style="list-style-type: none"> <li>S-Control Name</li> <li>Type</li> </ul>
<b>Static Resource</b>	<ul style="list-style-type: none"> <li>Description</li> <li>File</li> </ul>		<ul style="list-style-type: none"> <li>Name</li> </ul>
<b>Tab</b>	<ul style="list-style-type: none"> <li>Description</li> <li>Encoding</li> <li>Has Sidebar</li> <li>Height</li> <li>Label</li> <li>S-control</li> <li>Splash Page Custom Link</li> <li>Type</li> <li>URL</li> <li>Width</li> </ul>	<ul style="list-style-type: none"> <li>Tab Style</li> </ul>	<ul style="list-style-type: none"> <li>Tab Name</li> </ul>
<b>Translation</b>	<ul style="list-style-type: none"> <li>All attributes</li> </ul>		
<b>Validation Rule</b>	<ul style="list-style-type: none"> <li>Description</li> </ul>	<ul style="list-style-type: none"> <li>Active</li> </ul>	<ul style="list-style-type: none"> <li>Rule Name</li> </ul>


Component	Developer Editable	Subscriber and Developer Editable	Locked
	<ul style="list-style-type: none"> <li>Error Condition Formula</li> <li>Error Location</li> <li>Error Message</li> </ul>		
<b>Visualforce Component</b>	<ul style="list-style-type: none"> <li>API Version</li> <li>Description</li> <li>Label</li> <li>Markup</li> </ul>		<ul style="list-style-type: none"> <li>Name</li> </ul>
<b>Visualforce Page</b>	<ul style="list-style-type: none"> <li>API Version</li> <li>Description</li> <li>Label</li> <li>Markup</li> </ul>		<ul style="list-style-type: none"> <li>Name</li> </ul>
<b>Workflow Email Alert</b>		<ul style="list-style-type: none"> <li>Additional Emails</li> <li>Email Template</li> <li>From Email Address</li> <li>Recipients</li> </ul>	<ul style="list-style-type: none"> <li>Description</li> </ul>
<b>Workflow Field Update</b>	<ul style="list-style-type: none"> <li>Description</li> <li>Field Value</li> <li>Formula Value</li> </ul>	<ul style="list-style-type: none"> <li>Lookup</li> </ul>	<ul style="list-style-type: none"> <li>Name</li> </ul>
<b>Workflow Outbound Message</b>	<ul style="list-style-type: none"> <li>Description</li> <li>Endpoint URL</li> <li>Fields to Send</li> <li>Send Session ID</li> </ul>	<ul style="list-style-type: none"> <li>User to Send As</li> </ul>	<ul style="list-style-type: none"> <li>Name</li> </ul>
<b>Workflow Rule</b>	<ul style="list-style-type: none"> <li>Description</li> <li>Evaluation Criteria</li> <li>Rule Criteria</li> </ul>	<ul style="list-style-type: none"> <li>Active</li> </ul>	<ul style="list-style-type: none"> <li>Rule Name</li> </ul>
<b>Workflow Task</b>		<ul style="list-style-type: none"> <li>Assign To</li> <li>Comments</li> <li>Due Date</li> <li>Priority</li> <li>Record Type</li> <li>Status</li> </ul>	<ul style="list-style-type: none"> <li>Subject</li> </ul>

## Components Automatically Added to Packages

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

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

When you add this component	These components are automatically added
Action	Action target object (if it's a custom object), action target field, action record type, predefined field values, action layout; and any custom fields that the action layout or predefined values refer to on the target object
Reporting Snapshot	Reports
Apex class	Custom fields, custom objects, and other explicitly referenced Apex classes, and anything else that the Apex class references directly   <b>Note:</b> If an Apex class references a custom label, and that label has translations, you must explicitly package the individual languages desired for those translations to be included.
Apex trigger	Custom fields, custom objects, and any explicitly referenced Apex classes, and anything else that the Apex trigger references directly
Article type	Custom fields, the default page layout
Compact layout	Custom fields
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 fields, list views, page layouts, record types, validation rules, or custom buttons or links.
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   <b>Note:</b> <ul style="list-style-type: none"> <li>Apex sharing reasons are unavailable in extensions.</li> <li>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.</li> </ul>

When you add this component	These components are automatically added
Custom object (as an external object)	<p>External data source, custom fields, page layouts, list views, custom buttons, and custom links</p> <p> <b>Note:</b></p> <ul style="list-style-type: none"> <li>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.</li> <li>In managed and unmanaged packages, external objects are included in the custom object component.</li> </ul>
Custom tab	Custom objects (including all of its components), s-controls, and Visualforce pages
Dashboard	Folders, reports (including all of its components), s-controls, and Visualforce pages
Document	Folder
Email template (Classic)	<ul style="list-style-type: none"> <li>Folder</li> <li>Letterhead</li> <li>Custom fields</li> <li>Documents (stored as images on the letterhead or template)</li> </ul>
Email template (Lightning)	<ul style="list-style-type: none"> <li>Custom object</li> <li>Custom field references (in Handlebars Merge Language syntax)</li> <li>Enhanced folder (except the default public and private folders)</li> <li>Inline images referencing Salesforce Files</li> <li>Attachments referencing Salesforce Files</li> </ul> <p>For Lightning email templates created before Spring '21, attachments aren't automatically added to the package. Open and resave these templates to turn the attachments into content assets, which are then automatically added to the package</p> <p>These items aren't included and can't be added to a package:</p> <ul style="list-style-type: none"> <li>Enhanced letterhead</li> <li>The associated FlexiPage</li> <li>CMS files (Pardot only)</li> </ul>
Email template (Lightning) created in Email Template Builder	<ul style="list-style-type: none"> <li>Custom object</li> <li>Custom field references (in Handlebars Merge Language syntax)</li> <li>Enhanced folder (except the default public and private folders)</li> <li>Attachments referencing Salesforce Files</li> <li>The associated FlexiPage</li> </ul> <p>These items aren't included and can't be added to a package:</p> <ul style="list-style-type: none"> <li>Enhanced letterhead</li> <li>Inline images referencing Salesforce Files</li> <li>CMS files (Pardot only)</li> </ul>

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



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

## Special Behavior of Components in Packages

Building an app for distribution has special considerations and it's important to factor in 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 managed or unmanaged packages.

### 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're exposing any methods as Web services, include detailed documentation so that subscribers can write external code that calls your Web service.
- If an Apex class references a custom label and that label has translations, explicitly package the individual languages desired to include those translations in the package.
- If you reference a custom object's sharing object (such as `MyCustomObject__share`) in Apex, you add a sharing model dependency to your package. Set the default org-wide access level for the custom object to Private so other orgs can install your package successfully.
- The code contained in an Apex class, trigger, or Visualforce component that's part of a managed package is obfuscated and can't be viewed in an installing org. The only exceptions are methods declared as global. You can view global method signatures in an installing org. In addition, License Management Org users with the View and Debug Managed Apex permission can view their packages' obfuscated Apex classes when logged in to subscriber orgs via the Subscriber Support Console.
- You can use the `deprecated` annotation in Apex to identify `global` methods, classes, exceptions, enums, interfaces, and variables that can't be referenced in later releases of a managed package. So you can refactor code in managed packages as the requirements evolve. After you upload another package version as Managed - Released, new subscribers that install the latest package version can't see the deprecated elements, while the elements continue to function for existing subscribers and API integrations.
- Any Apex contained in an unmanaged package that explicitly references a namespace can't be uploaded.
- Apex code that refers to Data Categories can't be uploaded.

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

### Apex Sharing Reasons

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

### Connected Apps

- Connected apps can be added to managed packages, only. Connected apps are not supported for unmanaged packages.
- Subscribers or installers of a package can't delete a connected app by itself; they can only uninstall its package. A developer can delete a connected app after a package is uploaded as Managed - Released. The connected app is deleted in the subscriber's org during a package upgrade.
- To publish updates for a connected app that's part of a managed package, you typically push a new managed package version and upgrade subscriber orgs to the new version. But if you update a connected app's PIN Protect settings, it's not necessary to push a new managed package upgrade. After saving changes to PIN Protect settings, these updates are automatically published to subscriber orgs.
- The following connected app settings can't be updated with managed package patches.
  - Mobile App settings
  - Push messaging, including Apple, Android, and Windows push notifications
  - Canvas App settings
  - SAML settings

To update these settings, publish a new package version.

- If you push upgrade a package containing a connected app whose OAuth scope or IP ranges have changed from the previous version, the upgrade fails. This security feature blocks unauthorized users from gaining broad access to a customer org by upgrading an installed package. A customer can still perform a pull upgrade of the same package. This upgrade is allowed because it's with the customer's knowledge and consent.
- You can add an existing connected app (one created before Summer '13) to a managed package. You can also combine new and existing connected apps in the same managed package.
- For connected apps created before Summer '13, the existing install URL is valid until you package and upload a new version. After you upload a new version of the package with an updated connected app, the install URL no longer works.

### Custom Console

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

### Custom Fields

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

### Custom Labels

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

### Custom Metadata Types

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

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

### Custom Objects

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

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

### Custom Permissions

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

### Custom Report Types

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

### Custom Settings

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

### Custom Tabs

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

### Customer Portal and Partner Portal

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

### Dashboard Components

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

## Divisions

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

## Email Templates (Lightning)

Create managed packages using the first-generation packaging tool.

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

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

 **Note:** An email template created in Email Template Builder can't be edited after it's downloaded. To edit the template, clone it.

When upgrading a package that has Email Template Builder email templates, only the associated FlexiPage is updated. After downloading the new version of the template, clone it to see the changes.

## External Data Sources

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

## External Objects

- In managed and unmanaged packages, external objects are included in the custom object component.

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


### Field Dependencies

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

### Field Sets

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

If a package developer:	Then in the package upgrade:
Changes a field from <b>Unavailable</b> to <b>Available for the Field Set</b> or <b>In the Field Set</b>	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 <b>Available for the Field Set</b> or <b>In the Field Set</b> to <b>Unavailable</b>	The field is removed from the upgraded field set.
Changes a field from <b>In the Field Set</b> to <b>Available for the Field Set</b> (or vice versa)	The change isn't reflected in the upgraded field set.

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

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

### Flows

- When you upload a package or package version, the active flow version is included. If the flow has no active version, the latest version is packaged.
- To update a managed package with a different flow version, activate that version and upload the package again. Or deactivate all versions of the flow, make sure the latest flow version is the one to distribute, and then upload the package.
- In a development org, you can't delete a flow or flow version after you upload it to a released or beta managed package.
- You can't delete flows from Managed - Beta package installations in development org.
- You can't delete a flow from an installed package. To remove a packaged flow from your org, deactivate it and then uninstall the package.
- If you have multiple versions of a flow installed from multiple unmanaged packages, you can't remove only one version by uninstalling its package. Uninstalling a package—managed or unmanaged—that contains a single version of the flow removes the entire flow, including all versions.
- You can't include flows in package patches.

- An active flow in a package is active after it's installed. The previous active version of the flow in the destination org is deactivated in favor of the newly installed version. Any in-progress flows based on the now-deactivated version continue to run without interruption but reflect the previous version of the flow.
- Upgrading a managed package in your org installs a new flow version only if there's a newer flow version from the developer. After several upgrades, you can end up with multiple flow versions.
- If you install a managed package that contains multiple flow versions in a fresh destination org, only the latest flow version is deployed.
- If you install a flow from an unmanaged package that has the same name but a different version number as a flow in your org, the newly installed flow becomes the latest version of the existing flow. However, if the packaged flow has the same name and version number as a flow already in your org, the package install fails. You can't overwrite a flow.
- Flow Builder can't open flows that are installed from managed packages, unless they're templates.
- You can't create a package that contains flows invoked by both managed and unmanaged package pages. As a workaround, create two packages, one for each type of component. For example, suppose that you want to package a customizable flow invoked by a managed package page. Create one unmanaged package with the flow that users can customize. Then create another managed package with the Visualforce page referencing the flow (including namespace) from the first package.
- When you translate a flow from a managed package, the flow's Master Definition Name doesn't appear on the Translate page or the Override page. To update the translation for the Master Definition Name, edit the flow label and then update the translation from the Translate page.
- If any of the following elements are used in a flow, packageable components that they reference aren't included in the package automatically. To deploy the package successfully, manually add those referenced components to the package.
  - Post to Chatter
  - Send Email
  - Submit for Approval
- If a flow references a Lightning component that depends on a CSP Trusted Site, the trusted site isn't included in the package or change set automatically.

## Folders

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

## Home Page Components

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

## Home Page Layouts

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

### Inbound Network Connections

- Packaged connections are installed as unprovisioned. Alert subscribers about how to provision connections after package installation.
- If a developer changes the Region of a packaged connection that is subscriber-provisioned, the upgrade fails for the subscriber. Alert subscribers about tearing down the connection before updating the Region field. As a best practice, avoid changing the Region of a packaged connection unless necessary.

### List Views

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

### Multi-Currency

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

### Named Credentials

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

If you have multiple orgs, you can create a named credential with the same name but with a different endpoint URL in each org. You can then package and deploy—on all the orgs—one callout definition that references the shared name of those named credentials. For example, the named credential in each org can have a different endpoint URL to accommodate differences in development and production environments. If an Apex callout specifies the shared name of those named credentials, the Apex class that defines the callout can be packaged and deployed on all those orgs without programmatically checking the environment.

- Certificates aren't packageable. If you package a named credential that specifies a certificate, make sure that the subscriber org has a valid certificate with the same name.

### Outbound Network Connections

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

- If you package a Named Credential that references an Outbound Network Connection, the referenced Outbound Network Connection component is automatically added to the package.

### Page Layouts

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

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

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

### Picklist Values

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

### Record Types

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

### Reporting Snapshots

Developers of managed packages must consider the implications of introducing reporting snapshots that reference reports released in a previous version of the package. If the subscriber deleted the report or moved the report to a personal folder, the reporting snapshot referencing the report isn't installed, even though the Package Installation page indicates that it will be. Also, if the subscriber

has modified the report, the report can return results impacting the information displayed by the reporting snapshot. As a best practice, the developer releases the reporting snapshot and the related reports in the same version.

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

## Reports

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

- Hierarchy drill-downs are dropped from activity and opportunities reports.
- Filters on unpackageable fields are automatically dropped (for example, in filters on standard object record types).
- Package upload fails if a report includes filter logic on an unpackageable field (for example, in filters on standard object record types).
- Lookup values on the `Select Campaign` field of standard campaign reports are dropped.
- Reports are dropped from packages if they've been moved to a private folder or to the Unfiled Public Reports folder.
- When a package is installed into an org that doesn't have Chart Analytics 2.0:
  - Combination charts are downgraded instead of dropped. For example, a combination vertical column chart with a line added is downgraded to a simple vertical column chart. A combination bar chart with more bars is downgraded to a simple bar chart.
  - Unsupported chart types, such as donut and funnel, are dropped.

## S-Controls

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

S-controls have been deprecated and are superseded by [Visualforce](#) pages.

## Translation Workbench

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

## Validation Rules

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

## Analytics

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

- Analytics unmanaged packages, as opposed to managed packages, are considered a developer-only feature and aren't supported for general-purpose distribution. While Analytics unmanaged packages work as expected within the constraints of Salesforce unmanaged packages, they aren't subject to as much testing as managed packages. Unmanaged packages come without many of the safeguards of managed packages, and are intended for developers familiar with their limitations. Also refer to the relevant topic in the [ISV Guide](#).
- Before a recipe is available for packaging, you must create a dataset with the recipe. The related dataflow must be added to the package along with the recipe for deployment to succeed.

- Analytics Admin permissions are required to create a package but not for deployment, which requires only Salesforce admin permissions.
- There's no spidering between datasets and dataflows, meaning there's no dependency following. When packaging both, they must be added manually. If they aren't, an error appears during deployment. The same is true for change sets—when packaging both datasets and dataflows, add them manually.
- When you package a data flow, source and security predicates aren't included in the package.
- Because views are user-specific, they aren't included when you package the dashboard.
- If you migrate dashboards manually using JSON copy-and-paste, any conditional formatting, widget-specific number formats, and measure labels on blended queries are lost. To retain these formats and labels in the migrated dashboard, include the Analytics Dataset Metadata component type when packaging your change set.
- All custom fields used in a deployed dataflow must exist on the target org. The Analytics Cloud Integration User user must have access to the custom fields.
- Unlike Apex or other code, the underlying JSON definition of Analytics assets are visible in a managed package.
- The Winter '18 release contains a beta version of Apex steps, which lets developers include custom Apex functionality in a dashboard to access Salesforce platform features that aren't inherently supported in Analytics. If you include dashboards in a package, Apex steps aren't included—migrate Apex classes separately.
- Before the Spring '17 release, images didn't render when deploying a dashboard that used an image widget that referenced image files not available on the target org. There were two workarounds: Manually upload the images, or add a folder containing the images to the package. As of the Spring '17 release, images are packaged with the dashboard, and references between dashboards are maintained. You can't delete a dashboard that is referenced in a link. Either re-create the image, or link the widgets in the dashboard in the source org. Then repackage or fix the link issues in the target org.
- Take care when packaging dataflows. Invalid schema overrides and unsupported or illegal parameters are removed. For example, `Type = dim` is no longer supported. Use `Type = text` instead. Comments in JSON are removed. Nodes can appear in a different order.

## Workflow

- Salesforce prevents you from uploading workflow alerts that have a public group, partner user, or role recipient. Change the recipient to a user before uploading your app. During installation, Salesforce replaces that user with the user installing the app, and the installer can customize it as necessary.
- Salesforce prevents you from uploading workflow field updates that change an `Owner` field to a queue. Change the updated field value to a user before uploading your app. During installation, Salesforce replaces that user with the user installing the app, and the installer can customize it as necessary.
- Salesforce prevents you from uploading workflow rules, field updates, and outbound messages that reference a record type on a standard or managed-installed object.
- Salesforce prevents you from uploading workflow tasks that are assigned to a role. Change the `Assigned To` field to a user before uploading your app. During installation, Salesforce replaces that user with the user installing the app, and the installer can customize it as necessary.
- You can package workflow rules and associated workflow actions, such as email alerts and field updates. However, any time-based triggers aren't included in the package. Notify your installers to set up any time-based triggers that are essential to your app.  
  
Flow triggers aren't packageable. The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.
- Developers can protect some workflow actions.

- Developers can associate or disassociate workflow actions with a workflow rule at any time. These changes, including disassociation, are reflected in the subscriber's org upon install. In managed packages, a subscriber can't disassociate workflow actions from a workflow rule if it was associated by the developer.
- References to a specific user in workflow actions, such as the email recipient of a workflow email alert, are replaced by the user installing the package. Sometimes workflow actions referencing roles, public groups, account team, opportunity team, or case team roles aren't 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.


## Test and Respond to the New Order Save Behavior

If you created any type of package that includes the Order object, the installed package sometimes doesn't work, and package upgrades or new package installations are blocked. Here's why. The [Order Save Behavior Release Update](#) addresses an issue where Salesforce didn't correctly evaluate custom application logic on records associated with the Order object.

To ensure the expected behavior, you must test the Enable New Order Save Behavior release update. Starting in Winter '21, if a subscriber org relies on a different order save behavior than their installed packages, the installed packages sometimes don't work, and package upgrades or new package installations are blocked.

After the Enable New Order Save Behavior release update is enabled, Salesforce evaluates and runs these customizations whenever an update to an order item record changes the parent order record.

- Order and order item validation rules
- Order and order item Apex triggers and classes
- Order and order item workflow rules
- Order and order item flows and processes

 **Note:** The New Order Save Behavior release update affects all package types: unlocked, unmanaged, first-generation managed package (1GP), and second-generation managed package (2GP).

After you verify that your package works with the new order save behavior and that all your packages associated with your Dev Hub org work with the new order save behavior, you can either enable the release update in your Dev Hub org or wait for it to be auto-enabled in Summer '22. We recommend supporting both the new and old order save behavior during the Release Update window.

The scenarios in the table take effect in Winter '21 and continue through the release update window in Summer '22.

**Table 1: Success and Failure Scenarios for Packages and Order Save Behavior**

Package Uses	Subscriber Org Has Enabled New Order Save Behavior	Subscriber Org Has Enabled Old Order Save Behavior
New Order Save Behavior	<p>SUCCEED</p> <p>Package upgrades and installations of major and minor package versions succeed. All package types succeed in this condition.</p> <p>FAIL</p> <p><b>Winter '21</b></p> <p>Patch upgrades for 1GP packages fail. If you enabled new order save behavior,</p>	<p>FAIL</p> <p>If the subscriber org and the package are specifying different order save behaviors, package upgrades and installations are blocked.</p> <p>All package types fail in this condition.</p>

Package Uses	Subscriber Org Has Enabled New Order Save Behavior	Subscriber Org Has Enabled Old Order Save Behavior
	<p>create minor versions instead of patch versions of packages.</p> <p><b>Summer '21 and later</b></p> <p>Patch upgrades fail for 1GP packages that contain Salesforce Flow. If you enabled new order save behavior for packages with Salesforce Flow, create minor versions instead of patch versions of packages. All other patch upgrades for 1GP packages are successful.</p>	
Old Order Save Behavior	<p>FAIL</p> <p>If the subscriber org and the package are specifying different order save behaviors, package upgrades and installations are blocked.</p> <p>All package types fail in this condition.</p>	<p>SUCCEED</p> <p>New package installations and major, minor, and patch upgrades succeed.</p> <p>All package types succeed in this condition.</p>
Both New and Old Order Save Behavior	<p>SUCCEED</p> <p>New package installations of major and minor package version upgrades succeed.</p>	<p>SUCCEED</p> <p>New package installations of major and minor package version upgrades succeed.</p>

### Test Unmanaged and First-Generation Managed Packages

1. From Setup, in the Quick Find box, enter *Release Updates*, and then select **Release Updates**. Locate the Enable New Order Save Behavior tile, and select **Enable Test Run**.
2. Test the impact of the new behavior when an order or order item is edited. Review any custom application logic such as validation rules, Apex triggers and classes, workflow rules, flows, and processes.  
We recommend supporting both the new and old order save behavior during the Release Update window.
3. To indicate that your package is compatible with both new and old order save conditions, from Setup, in the Quick Find box, enter *Package*. Select the package that you tested and select **Upload**.
4. Locate the Package Requirements section and disable **New Order Save Behavior**.  
When this setting is disabled and the release update is enabled, subscriber orgs using either the new or old order save behavior can install your package.

### Test Unlocked and Second-Generation Managed Packages

1. After creating a scratch org, enable the Release Update in it. From Setup, in the Quick Find box, enter *Release Updates*, and then select **Release Updates**. Locate the Enable New Order Save Behavior tile, and select **Enable Test Run**.
2. Test the impact of the new behavior when an order or order item is edited. Review any custom application logic such as validation rules, Apex triggers and classes, workflow rules, flows, and processes.

When you're ready to create a package version, specify the order save behavior in the definition file.

Table 2: Order Save Behavior Options

To Specify	Set Features in Scratch Org Definition File To
Old Order Save Behavior	<pre>{   "features": [],   "settings": {     "orderSettings": {       "enableOrders": true     }   } }</pre>
New Order Save Behavior	<pre>{   "features": ["OrderSaveLogicEnabled"],   "settings": {     "orderSettings": {       "enableOrders": true     }   } }</pre>
New and Old Order Save Behavior	<pre>{   "features": ["OrderSaveBehaviorBoth"],   "settings": {     "orderSettings": {       "enableOrders": true     }   } }</pre>

## Protected Components

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

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

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


- Workflow flow triggers

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

## Set Up a Platform Cache Partition with Provider Free Capacity

Salesforce provides 3 MB of free Platform Cache capacity for AppExchange-certified and security-reviewed managed packages. This is made available through a capacity type called Provider Free capacity and is automatically enabled in all Developer edition orgs.

Follow the steps here to allocate the Provider Free capacity to a Platform Cache partition before adding it to your managed package.

 **Note:** If a Platform Cache partition is already part of your managed package, you can choose to edit the existing partition and allocate the Provider Free capacity to it.


Create a partition from the Platform Cache page and then set it up to use the Provider Free capacity

1. From Setup, in the Quick Find box, enter *Platform Cache*, and then select **Platform Cache**.


As the Provider Free capacity is automatically enabled in all Developer edition orgs, the Org's Capacity Breakdown donut chart shows the Provider Free capacity.

2. Click **New Platform Cache Partition**.
3. In the **Label** box, enter a name for the partition. The name can contain alphanumeric characters only and must be unique in your org.
4. In the **Description** box, enter an optional description for the partition.
5. In the Capacity section, allocate separate capacities for session cache and org cache from the available Provider Free capacity.
6. Save the new Platform Cache partition.

You can add this new Platform Cache partition to your managed package. When an AppExchange-certified, security-reviewed managed package with Platform Cache partition is installed on the subscriber org, the Provider Free capacity is allocated and automatically made available to the installed partition. The managed package can start using the Platform Cache partition; no post-install script or manual allocation is required.

 **Note:** If the managed package is not AppExchange-certified and security-reviewed, the Provider Free capacity resets to zero and will not be allocated to the installed Platform Cache partition.

When a Platform Cache partition with Provider Free capacity is installed in a subscriber org, the Provider Free capacity allocated is non-editable. The provider free capacity of one installed partition can't be used for any other partition.

 **Tip:** After you install a Platform Cache partition with Provider Free capacity, you can edit the partition and make additional allocations from the available platform cache capacity of the org.


## 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:
  - 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

### 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 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 to use permission sets or profile settings alone or in combination, consider the similarities and differences.

Behavior	Permission Sets	Profile Settings
What permissions and settings are included?	<ul style="list-style-type: none"><li>• Assigned custom apps</li><li>• Custom object permissions</li><li>• External object permissions</li><li>• Custom field permissions</li><li>• Custom metadata types permissions</li><li>• Custom permissions</li><li>• Custom settings permissions</li><li>• Custom tab visibility settings</li><li>• Apex class access</li><li>• Visualforce page access</li><li>• External data source access</li><li>• Record types</li></ul>	<ul style="list-style-type: none"><li>• Assigned custom apps</li><li>• Assigned connected apps</li><li>• Tab settings</li><li>• Page layout assignments</li><li>• Record type assignments</li><li>• Custom field permissions</li><li>• Custom metadata type permissions</li><li>• Custom object permissions</li><li>• Custom permissions</li><li>• Custom settings permissions</li><li>• External object permissions</li><li>• Apex class access</li></ul>

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

Behavior	Permission Sets	Profile Settings
	<p>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.</p> <p>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.</p>	
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.

## Permission Set Groups

You can organize permission sets into groups and include them in first and second-generation managed packages. Permission set groups can be updated when you upgrade the package.

Keep these considerations in mind when you organize permission sets into groups to include in your managed packages:

 **Important:** You can't include object permissions for standard objects in managed packages. During package installation, all object permissions for standard objects are ignored, and aren't installed in the org.

Also:

- You can't add permission sets constrained by a permission set license to managed or unmanaged packages.
- You can only package permissions for metadata that's included in your package.

SEE ALSO:

[Create a Permission Set Group](#)

[Permission Set Groups Considerations](#)

## 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 enabled, 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 and Visualforce URLs

Salesforce org application URLs vary based on multiple configuration options. Deploying a My Domain updates your Visualforce, Experience Builder, and content URLs. A My Domain setting can further alter the format of those URLs. Enabling enhanced domains affects almost all application URL formats, including URLs for Experience Cloud sites and Salesforce Sites. It changes domain suffixes (the static part at the end of the URLs) and removes instance names from all URLs. To build packages that support all the possible URL formats, replace hard-coded URL references with relative URLs.

Because enhanced My Domains meet the latest browser requirements, they're the future standard. However, until they're enabled in all Salesforce orgs, packages must accommodate all possible URL formats. If your package includes URLs, use relative URLs to avoid any issues with domain names. Otherwise, users can have page functionality issues when using outdated packages.

For example, there are four possible Visualforce URL formats:

- `PackageName.InstanceName.visual.force.com`
- `MyDomainName--PackageName.visualforce.com`
- `MyDomainName--PackageName.InstanceName.visual.force.com`
- `MyDomainName--PackageName.vf.force.com`

To account for all these formats, if the URL for one of your org's Visualforce pages is

`MyDomainName--PackageName.vf.force.com/{Case.Id}`, use the relative path when referencing the page:  
`{Case.Id}`.

For more information on enhanced domains, see [Enhanced Domains](#) in *Salesforce Help*. For information on the various login and application URLs for Salesforce orgs, see [My Domain URL Formats](#) in *Salesforce Help*.

## Developing App Documentation

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

### 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.

### EDITIONS

Available in: Salesforce Classic ([not available in all orgs](#))

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

**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.

There are two possible options for the **API Access** setting:

- **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.

### EDITIONS

Available in: Salesforce Classic ([not available in all orgs](#))

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

## 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.

### 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 79.

#### 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.

#### 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

## 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 79.

### 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 52.0, 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.

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 , 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.

### EDITIONS

Available in: Salesforce Classic

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

### USER PERMISSIONS

To download a WSDL:

- Customize Application

- 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 allowlist a connected app to use 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 allowlisting.
- You can enable SOAP-based Web services, including Metadata API, using an API token called a Client ID. Append the Client ID 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 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 allowlisting
Apex methods exposed as Web services (REST)	Yes, with connected app consumer allowlisting
Connect 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 Trailforce 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

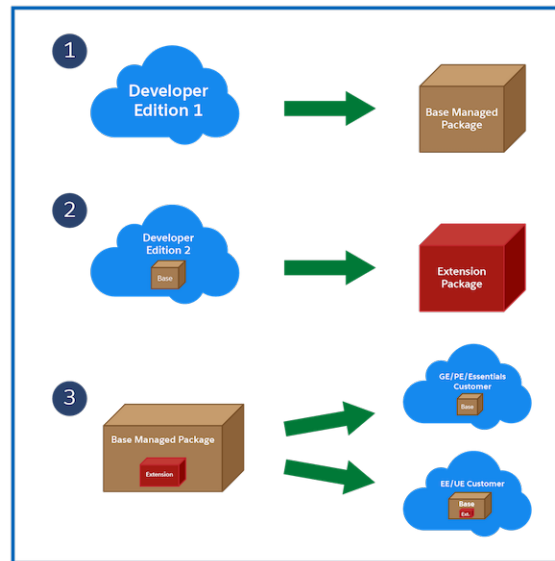
Supporting multiple editions provides the opportunity to release richer versions of your app that can support more advanced features found in EE, UE, and PXE. There are two technologies that can be leveraged to support multiple editions. The first approach uses extension packages and the second leverages 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 you 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

---

A connected app is a framework that enables an external application to integrate with Salesforce using APIs and standard protocols, such as SAML, OAuth, and OpenID Connect. Connected apps use these protocols to authenticate, authorize, and provide single sign-on (SSO) for external apps. The external apps that are integrated with Salesforce can run on the customer success platform, other platforms, devices, or SaaS subscriptions. For example, when you log in to your Salesforce mobile app and see your data from your Salesforce org, you're using a connected app.

By capturing metadata about an external app, a connected app tells Salesforce which authentication protocol—SAML, OAuth, and OpenID Connect—the external app uses, and where the external app runs. Salesforce can then grant the external app access to its data, and attach policies that define access restrictions, such as when the app's access expires. Salesforce can also audit connected app usage.

To learn more about how to use, configure, and manage connected apps, see the following topics in *Salesforce Help*:

- [Connected App Use Cases](#)
- [Create a Connected App](#)
- [Edit a Connected App](#)
- [Manage Access to a Connected App](#)

## More Resources

Here are some additional resources to help you navigate connected apps:

- *Salesforce Help*: [Connected Apps](#)
- *Salesforce Help*: [Authorize Apps with OAuth](#)
- *Trailhead*: [Build Integrations Using Connected Apps](#)

## 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.

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.

### EDITIONS

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

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

### [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 Salesforce 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 <i>Service Provider [Organization ID]</i> , where <i>Organization ID</i> is the ID of the member org. Users who don't have access to the service provider sometimes see this message when attempting

Profile Section	Environment Hub Settings
	<p>to log in via SSO: "User '[UserID]'" does not have access to sp '[Service Provider ID]'."</p> <p>When configuring the Environment Hub in a new org, this section is empty.</p>
Administrative Permissions	<p>Enable "Manage Environment Hub" to allow users to:</p> <ul style="list-style-type: none"> <li>• Create orgs for development, testing, and trials.</li> <li>• Configure SSO for member orgs.</li> </ul>
General User Permissions	<p>Enable "Connect Organization to Environment Hub" to allow users to connect existing orgs to the Environment Hub.</p>
Standard Object Permissions	<p>Grant object permissions based on the level of access required by the Environment Hub user.</p> <p>Hub Members object:</p> <ul style="list-style-type: none"> <li>• "Read"—View existing Hub Member records.</li> <li>• "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.</li> <li>• "Edit"—Edit fields on existing Hub Member records.</li> <li>• "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).</li> <li>• "View All"—Read all Hub Member records, regardless of who created them.</li> <li>• "Modify All"—Read, edit, and delete all Hub Member records, regardless of who created them.</li> </ul> <p>Hub Invitations object:</p> <ul style="list-style-type: none"> <li>• If you enable the "Connect Organization to Environment Hub" permission, enable "Create", "Read", "Update, and "Delete" for Hub Invitations.</li> </ul> <p>Signup Request object:</p> <ul style="list-style-type: none"> <li>• 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.</li> </ul>

d. Select **Save**.

## Enable My Domain for the Environment Hub

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

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

1. [Set Up My Domain](#). Choose or change your My Domain name, then update your org to use your new URLs.



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

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

### EDITIONS

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

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

### USER PERMISSIONS

To set up My Domain:

- Customize Application

To define a My Domain name:

- Customize Application  
AND  
Modify All Data

## Manage Orgs in the Environment Hub

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

### [Connect an Org to the Environment Hub](#)

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

### [Create an Org from the Environment Hub](#)

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

### 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 (except scratch orgs) from one location. When you connect an org to the hub, related orgs are automatically discovered so you don't have to manually connect them.

The following types of related orgs are automatically discovered.

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

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

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

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

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


### USER PERMISSIONS

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

- Connect Organization to Environment Hub

## Create an Org from the Environment Hub

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

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

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

### USER PERMISSIONS

To set up and configure the Environment Hub:

- Manage Environment Hub

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 <a href="https://www.salesforce.com/trial">www.salesforce.com/trial</a> . When you create a Test/Demo org, you can specify a Trialforce template if you want the org to include your customizations.

Purpose	Lets You Create:
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.

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**.

### USER PERMISSIONS

To set up and configure the Environment Hub:

- Manage Environment Hub

## Use a Federation ID or Formula for SSO

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

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

Method	Steps
SSO Method 2 - Federation ID	Select the checkbox.
SSO Method 3 - User Name Formula	Select the checkbox, and then define a formula. For example, to match the first part of the username (the

### USER PERMISSIONS

To set up and configure the Environment Hub:

- Manage Environment Hub

Method	Steps
	part before the "@" sign) with an explicit domain name, enter: <pre>LEFT(\$User.Username, FIND("@", \$User.Username)) &amp; ("mydev.org")</pre>

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 Salesforce Source Organizations created from an Environment Hub that's also a Salesforce 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?](#)

[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?](#)

### EDITIONS

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

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

## 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 Salesforce Source Organizations created from an Environment Hub that's also a Salesforce Management Organization.

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

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

## Can I enable the Environment Hub in a sandbox org?

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

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

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

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

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

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

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

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

## Can an org be a member of multiple Environment Hubs?

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

1. Go to the Environment Hub tab.
2. Find the org, from the drop-down select **Remove**.

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

## Can I disable the Environment Hub?

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

## Considerations for the Environment Hub in Lightning Experience

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

### List View Limitations

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

#### EDITIONS

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

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

## 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 command-line interface (CLI) to create scratch orgs.



**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.

#### EDITIONS

Available in: Lightning Experience

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

### SEE ALSO:

[Salesforce CLI Setup Guide](#)

[Salesforce DX Developer Guide](#)

## Scratch Org Allocations for Partners

To ensure optimal performance, 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.

### Summit Tier

- 300 active
- 600 daily

### Crest Tier

- 150 active
- 300 daily

### Ridge Tier

- 80 active
- 160 daily

### Base Tier

- 40 active
- 80 daily

### Partner Trials


- 20 active
- 40 daily

## Enable Dev Hub Features in Your Org

Enable Dev Hub features in your Salesforce org so you can create and manage scratch orgs, create and manage second-generation packages, and use Einstein features. Scratch orgs are disposable Salesforce orgs to support development and testing.


It's not necessary to enable Dev Hub if you plan to use Salesforce CLI with only sandboxes unless you plan to create second-generation (2GP) packages. The 2GP packages use a scratch org during the package generation process.

Enabling Dev Hub in a production or business org is safe and doesn'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.

 **Note:** You can't enable Dev Hub in a sandbox.

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

- You can create up to six scratch orgs and package versions per day, with a maximum of three 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.

 **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 (for customers), or your business org (for ISVs).
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 that 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.

### 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

## 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 create scratch orgs and use other developer-specific features.

You can use Salesforce DX with these standard user licenses: Salesforce, Salesforce Platform, and Developer.

If your org has Developer licenses, you can add users with the Developer profile and assign them the provided Developer permission set. Alternatively, you can add users with the Standard User or System Administrator profiles. For a standard user, you must create a permission set with the required Salesforce DX permissions. We recommend that you avoid adding users as system administrators unless their work requires that level of authority and not just Dev Hub org access.

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 aren't accessible via this license.



**Important:** Contact your Salesforce account executive to request the Free Limited Access License.

To use Org Shape for Scratch Orgs or Scratch Org Snapshots (pilot), be sure to assign the Salesforce or Salesforce Platform user license. This license isn't supported at this time.

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](#). Salesforce administrators can upgrade a Salesforce Limited Access - Free license to a standard Salesforce license at any time.



**Note:** This license doesn't provide access to some Salesforce CLI commands, such as `force:limits:api:display`. Contact your Salesforce admin for API limits information.

## 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.

 **Tip:** Make sure your browser allows pop-ups from your Dev Hub org.

- a. (Required) If you haven't already done so, define and deploy a My Domain name.

Production orgs created in Winter '21 and later have a My Domain by default. For information on setting up and deploying a My Domain, see [My Domain](#) in Salesforce Help.

 **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.

- b. From the App Launcher menu, select **Namespace Registries**.

- c. Click **Link Namespace**.

If you don't see the **Link Namespace** button, make sure your My Domain is deployed to users.

- d. From Setup, enter *My Domain* in the Quick Find box, then select **My Domain**.

- e. Do you see Step b? That means you registered a My Domain, but haven't deployed it yet. Search and go to **My Domain** in Quick Find. Click **Deploy to Users** (if it's not there, then it is deployed).

Log out of your Dev Hub org, then open it again.

 **Important:** Deploying or renaming a **My Domain** may take up to an hour to take effect.

2. Log in to the Developer Edition org in which your namespace is registered using the org's System Administrator's credentials.

You cannot link orgs without a namespace: sandboxes, scratch orgs, patch orgs, and branch orgs require a namespace to be linked 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

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:
```

### EDITIONS

Available in: Salesforce  
Classic

```
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

The screenshot shows the 'Package Edit' interface in Salesforce. The package name is 'Expense Mgmt Package' and it is marked as 'Managed'. The 'Language' is set to 'English'. The 'Notify on Packaging Error' field is highlighted with a red oval and contains the email address 'admin@xyz.org'. The 'Description' field is empty. The 'Created By' field shows 'Admin User' on '6/10/2016 9:52 AM' and the 'Modified By' field shows 'Admin User' on '7/27/2016 9:15 AM'. There are 'Save' and 'Cancel' buttons at the top and bottom of the form.


## 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

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

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

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

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

### EDITIONS

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

Available in: **Developer Edition**

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

## Configure Your Developer Settings

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

To configure your developer settings:

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



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

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



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

## Register a Namespace Prefix

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



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

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

### EDITIONS

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

Available in: **Developer Edition**

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

### USER PERMISSIONS

To configure developer settings:

- Customize Application

To create packages:

- Create AppExchange Packages

To upload packages:


- Upload AppExchange Packages

### EDITIONS

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

Available in: **Developer Edition**

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

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


Your namespace prefix must:

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

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

To register a namespace prefix:

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

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

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

## 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>.

## 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.

### 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**

- 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 123 and [Installing a Package](#) on page 120.

## 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 125.
  - 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 129.
  - 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 38. 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 55.

- 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 72. 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

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

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 111. Your developer settings specify your [namespace prefix](#) on page 111, 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 113.

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.

### 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

- 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 125.
  - 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 129.
  - 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 38. 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 55.
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 72. 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 352.
  - 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 117
- [Components](#) on page 119
- [Versions](#) on page 119
- [Patch Organizations](#) on page 120

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 <b>Unrestricted</b> , which means that all package components that access the API have the same access as the user who is logged in. Click <b>Enable Restrictions</b> or <b>Disable Restrictions</b> 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	<p>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.</p> <p> <b>Note:</b> Apex can only be packaged from Developer, Enterprise, Unlimited, and Performance Edition organizations.</p>
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 <a href="#">Running Apex on Package Install/Upgrade</a> on page 125.
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 <a href="#">Running Apex on Package Uninstall</a> on page 129.

## 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 55.

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 72. 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 362.



**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	<p>Lists the actions you can perform on the package. The possible actions are:</p> <ul style="list-style-type: none"> <li>• <b>Deprecate:</b> Deprecates a package version. <ul style="list-style-type: none"> <li>•  <b>Warning:</b> Users can no longer download or install this package. However, existing installations continue to work.</li> </ul> </li> <li>• <b>Undeprecate:</b> Enables a previously deprecated package version to be installed again.</li> </ul>
Status	<p>The status of the package. The possible statuses are:</p> <ul style="list-style-type: none"> <li>• Released: The package is Managed - Released.</li> </ul>

Attribute	Description
	<ul style="list-style-type: none"> <li>• Beta: The package is Managed - Beta.</li> <li>• Deprecated: The package version is deprecated.</li> </ul>
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 <code>Version Number</code> .
Version Number	The version number for the latest installed package version. The format is <code>majorNumber.minorNumber.patchNumber</code> , such as 2.1.3. The version number represents a release of a package. The <code>Version Name</code> is a more descriptive name for the release. The <code>patchNumber</code> is generated only when you create a patch. If there is no <code>patchNumber</code> , 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 354. 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	<p>Lists the actions you can perform on a patch development organization. The possible actions are:</p> <ul style="list-style-type: none"> <li>• <b>Login:</b> Log in to your patch development organization.</li> <li>• <b>Reset:</b> Emails a new temporary password for your patch development organization.</li> </ul>
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

---

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.
  - When an installed package includes a custom field that's referenced by Einstein Prediction Builder or Case Classification, Salesforce prevents you from uninstalling the package. Before uninstalling the package, edit the prediction in Prediction Builder or Case Classification so that it no longer references the custom field.
- 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: **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
<code>compareTo</code>	<code>System.Version version</code>	Integer	<p>Compares the current version with the specified version and returns one of the following values:</p> <ul style="list-style-type: none"> <li>• Zero if the current package version is equal to the specified package version</li> <li>• An Integer value greater than zero if the current package version is greater than the specified package version</li> <li>• An Integer value less than zero if the current package version is less than the specified package version</li> </ul> <p>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.</p>
<code>major</code>		Integer	Returns the major package version of the calling code.
<code>minor</code>		Integer	Returns the minor package version of the calling code.
<code>patch</code>		Integer	Returns the patch package version of the calling code or <code>null</code> 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 ...

- [AppExchange Security Review](#)
- [How Does AppExchange Security Review Work?](#)
- [Partner Security Portal](#)
- [Test Your Entire Solution](#)
- [False Positives](#)
- [Security Review Resources](#)

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

## AppExchange Security Review

---

Before you can publicly list your solution on AppExchange, the solution must pass a security review. The AppExchange security review tests the security posture of your solution, including how well it protects customer data.

The security review helps you identify security vulnerabilities that a hacker, malware, or other threat can exploit. Salesforce security review teams test your solution with threat-modeling profiles that are based on the most common web vulnerabilities. The teams attempt to penetrate the defenses programmed in your solution. Their goal is to extract or modify data that they don't have permission to access, just as security threats attempt to do.


Here is a small sampling of the common security threats that we test for.

- SOQL and SQL injection
- Cross-site scripting
- Nonsecure authentication and access control protocols
- Vulnerabilities specific to the Salesforce platform, such as record-sharing violations

For more information about the most critical web application security risks, read the [Open Web Application Security Project \(OWASP\) Top Ten](#) awareness document. OWASP is a nonprofit foundation that works to improve the security of software.

The scope of cyberthreats is large, and Salesforce upholds high security standards for solutions distributed on AppExchange. The review process is rigorous. Don't get discouraged if your AppExchange submission isn't approved the first time, or if it requires multiple code changes. Many solutions don't pass security review the first time they're submitted.

We give you a report documenting the security vulnerabilities found during the review. We're also available to meet with you and help you address vulnerabilities. Address the issues in the report, then submit the revised solution for a follow-up review. We offer multiple reviews for each submission, which enables you to fine-tune the security of your solution.

 **Important:** To ensure that upgraded solutions safeguard against the latest security vulnerabilities, Salesforce reserves the right to conduct periodic re-reviews of solutions distributed on AppExchange.

View the security review process as enforcement mechanisms paired with personalized advice and tools. You have access to office hours where you can directly connect with a security review team member to get guidance catered to your solution. And, the security review team points you to security-scanning tools that help automate the process of vetting the security of your solution.


SEE ALSO:

[Open Web Application Security Project \(OWASP\) Top Ten](#)

## How Does AppExchange Security Review Work?

---

Before initiating an AppExchange security review, you perform your own testing and gather supporting materials that help us assess the security of your solution. During a review, our Product Security team attempts to identify security vulnerabilities in your solution. If the team identifies vulnerabilities, you have access to personalized technical guidance to help you address the identified vulnerabilities.

 **Important:** Every version of your managed package that you plan to list publicly on AppExchange must go through a security review. The review for a new version of a package that passed a security review is automated, and typically only takes a few minutes.



## Ensure That You're Ready to Start

Knowing when you're ready for a security review is as important as how it works. You're ready to submit a solution for security review after you:

- Confirm with a partner recruitment representative that your solution is enrolled in the AppExchange Partner Program, and that you have a distribution agreement.
- Secure your solution according to industry best security standards.
- Certify that your solution is [Lightning Ready](#). All new solutions submitted for security review must be Lightning Ready.
- In the Salesforce Partner Community Publishing Console:
  - Connect your packaging organization to AppExchange.
  - Create a provider profile.
  - Create a solution listing.
  - Submit a business plan for review, and receive Salesforce approval.

## Test Your Solution

Run automated scanning tools and manually test your solution throughout the solution development lifecycle. Security scanning tools provide only first-pass, though useful, insights into solution vulnerabilities. To find vulnerabilities that automated scanning tools don't detect, also manually test your solution.



**Tip:** We strongly recommend that you test your code throughout the development lifecycle. If you defer testing and remediation, you're likely to encounter a larger accumulation of issues and greatly delay your time to market.

After you finish developing your solution, perform another round of manual testing and run the automated scanning tools that Product Security requires. The type of scans that you're required to run depends on the architecture of your solution.

On the [Partner Security Portal](#), you can access the Source Code Scanner, which is also referred to as the Checkmarx scanner, and the Chimera scanner. These two scanning tools meet the test requirements for many AppExchange solutions.

Before you submit your solution for review, address all security issues that you find with your manual testing and the scanning tools. Either fix the code or document how flagged issues are false positives. A false positive is an issue that appears to pose a security risk but does not.

Test your solution before you submit it and you're much more likely to pass the review the first time. Applicants who don't test beforehand rarely pass and must resubmit after addressing security vulnerabilities identified during a review. Resubmitting significantly delays the solution publishing process.

## Gather the Required Materials for Security Review Submission

Assemble the materials that enable Product Security to perform a thorough manual review. For most submissions, you're required to provide a Developer Edition org with the version of the solution that you intend to distribute installed and solution documentation. The security review team uses the Developer Edition org as the solution test environment. The org and documentation aren't required for Marketing Cloud apps. Other required materials vary by solution type.



**Tip:** You're likely to have questions as you prepare for security review and at other points during or after a security review. To discuss your concerns and get answers to your questions, visit the [Partner Security Portal](#) and schedule an office hours appointment. For help with submitting your solution for review, schedule an appointment with the Security Review Operations team. To troubleshoot issues in your solution that were identified during a review, make an appointment with the Product Security team.

## Submit Your Solution for Review

After you complete testing and gather the materials required for your submission, you're ready to submit your solution for an AppExchange security review. Use the security review submission interface to share your solution and required materials, and to pay the security review and annual AppExchange listing fees. If you plan to distribute your solution for free, you don't pay the fees.

After you submit everything, expect these turnaround times.

Security Review Stage	Typical Time Frame
Security Review Operations verifies that your submission is ready to review. A submission is ready to review if it includes everything required to test the security of your solution.	1–2 days
Product Security tests your solution for the first time.	4–6 weeks
Product Security tests a resubmission of a package that wasn't approved previously and that shows progress in fixing security vulnerabilities.	2–3 weeks

## Follow Up on the Security Review Report

When the security review is complete, you receive a report informing you that your submission is approved or not approved for public listing on AppExchange.

- **Approved:** You can publicly list your solution on AppExchange and distribute it to customers immediately.
- **Not Approved:** The security review team detected security issues in your solution. You can't list your solution on AppExchange or distribute it to customers.

If your solution isn't approved, the report includes information about the types of security issues that we detected. Keep in mind that the security review is a black-box, time-limited process. We can't list every instance of a security issue, and we may not initially detect all issue types. Interpret the security review findings as representative examples of the types of issues you must fix. Then diligently find and fix all instances of each issue across your entire solution.

Address all detected security issues. Rerun the required automated scanning tools to generate reports for your revised solution. Then resubmit your revised solution with the updated scan reports.

#### SEE ALSO:

[Connect a Packaging Org to the Publishing Console](#)

[Create or Edit Your Provider Profile](#)

[Create or Edit Your AppExchange Listing](#)

[Add a Business Plan to an AppExchange Listing](#)

[Partner Security Portal](#)

[Partner Security Portal site](#)

[Lightning Ready for AppExchange Partners \(ISV\)](#)

## Partner Security Portal

The Partner Security Portal is the main hub for your security review needs. The portal hosts the Source Code Scanner (Checkmarx) and Chimera automated security scanning tools. Use these tools to identify security vulnerabilities in your solution. The portal is also where you go to schedule office hours appointments with AppExchange security engineers and Security Review Operations team members. Office hours provide a forum for you to ask questions about the security review process and to discuss how to rework code that has security vulnerabilities.

#### [Set Up Your Partner Security Portal Login](#)

Connect your packaging org to your Partner Community account on the Organizations tab of the Salesforce Partner Community publishing console. Then log in to the Partner Security Portal using that org's credentials. Logged in users can access security scanning tools and schedule office hours appointments.

#### [Security Scanners on the Portal](#)

To identify security vulnerabilities, we require that you run security scanning tools on your solution and all external endpoints that run independently of the Salesforce platform. The Partner Security Portal hosts two of the scanners that we recommend, the Source Code Scanner (Checkmarx) and Chimera.

#### [Office Hours Appointments on the Portal](#)

Salesforce security review teams host office hours for AppExchange partners. During office hours, you have direct, scheduled, web conference access to security review team members. Get answers about the submission process from Security Review Operations or troubleshoot security vulnerabilities with Product Security.

## Set Up Your Partner Security Portal Login

Connect your packaging org to your Partner Community account on the Organizations tab of the Salesforce Partner Community publishing console. Then log in to the Partner Security Portal using that org's credentials. Logged in users can access security scanning tools and schedule office hours appointments.



**Note:** Before you set up your Partner Security Portal login, ensure that the packaging org that hosts your development work is a Salesforce Developer Edition org.

1. Log in to the [Salesforce Partner Community](#).
2. Click **Publishing**.

#### USER PERMISSIONS

To access Source Code Scanner (Checkmarx) on the Partner Security Portal:

- Author Apex

3. Click **Organizations**.
4. Click **Connect Org**.
5. Enter the credentials that you use for your packaging org, and then log in.
6. Go to the [Partner Security Portal](#).
7. Click **Login**.

## Security Scanners on the Portal

To identify security vulnerabilities, we require that you run security scanning tools on your solution and all external endpoints that run independently of the Salesforce platform. The Partner Security Portal hosts two of the scanners that we recommend, the Source Code Scanner (Checkmarx) and Chimera.



**Tip:** We strongly recommend that you run security scans on your code and any connected endpoints throughout the development lifecycle. Run periodic scans and fix flagged issues as you go to prevent security vulnerabilities from piling up and creating more work for you later.

The [Partner Security Portal](#) provides access to two Salesforce-supported scanners: the Source Code Scanner, also referred to as the Checkmarx scanner, and the Chimera scanner service.

The Source Code Scanner (Checkmarx) checks Apex, Visualforce, and Lightning code, but doesn't check external endpoints of a solution.

Chimera checks external endpoints, but requires you to upload a token to the root of the external server. If your solution connects to endpoints on domains that you own, you can use Chimera. If your solution connects to endpoints on domains that you don't own, you can't upload the token and can't use Chimera. Use an alternative tool. For example, download the free OWASP Zed Attack Proxy (ZAP) scanner or purchase a license for Burp Suite.

Just before you submit your solution, except for mobile clients and API solutions, run the Source Code Scanner in the Partner Security Portal. If your solution connects to any non-Salesforce domains, also run Chimera, OWASP ZAP, or Burp Suite on the external endpoints. Include reports from your scans when you submit your solution for security review.

### USER PERMISSIONS

To access Source Code Scanner (Checkmarx) on the Partner Security Portal:

- Author Apex

Security Scanner	Scan Targets	Considerations
Source Code Scanner (Checkmarx)	Apex, Visualforce, and Lightning code	<ul style="list-style-type: none"> <li>• This static scanning tool uses Checkmarx security technology.</li> <li>• Mandatory for any security review submission that includes a Salesforce package or component. Not required for mobile clients or API solutions.</li> <li>• You're provisioned three Source Code Scanner (Checkmarx) runs per solution version with the security review fee. Consider running an alternative tool as you develop, such as the open-source PMD Source Code Analyzer, and the Source Code Scanner as you finalize your submission.</li> <li>• If you want the flexibility and freedom to scan unpackaged code, or bypass scan limits and package linking requirements, purchase a license from Checkmarx.</li> </ul>
Chimera	External endpoints on domains that you own	<ul style="list-style-type: none"> <li>• Checks for security vulnerabilities in external endpoints of a solution.</li> <li>• Scans solutions from a Salesforce IP address.</li> <li>• Doesn't require a download.</li> </ul>

Security Scanner	Scan Targets	Considerations
		<ul style="list-style-type: none"> <li>• Isn't usable with endpoints on domains that you don't own because it requires upload of a token to the root of the external server.</li> <li>• If your solution connects to external endpoints that you don't own, use OWASP ZAP or Burp Suite instead of Chimera.</li> </ul>

## SEE ALSO:

[Test Your Entire Solution](#)[OWASP Zed Attack Proxy \(ZAP\)](#)[Burp Suite](#)

## Office Hours Appointments on the Portal

Salesforce security review teams host office hours for AppExchange partners. During office hours, you have direct, scheduled, web conference access to security review team members. Get answers about the submission process from Security Review Operations or troubleshoot security vulnerabilities with Product Security.



**Note:** To schedule an office hours appointment, visit the [Partner Security Portal](#). Need a portal login? Follow the instructions in [Set Up Your Partner Security Portal Login](#).

## Operations Office Hours

During operations office hours, Security Review Operations team members answer questions about security review logistics and submission requirements. Typical questions include:

- What components of the solution are in scope for the security review?
- What type of reports and scan results am I required to provide?
- What are the fees for posting a paid solution on AppExchange?
- What is the status of my security review?
- When do I receive the results for my submission?
- What happens if the solution that I submit doesn't pass the review?

## Technical Office Hours

Technical office hours are available when you need specific security-related technical assistance from the Product Security team. Typical questions include:

- How do I navigate the AppExchange security requirements?
- What is a secure way to design and implement a specific aspect of my solution?
- How do I address issues that the automated security scanning tools detect?
- What does a finding in my security review report mean?
- What security scan results can I regard as false positives?
- How do I resolve the issues in my security review report that I think are false positives?
- Does my reworking of the code fix the security vulnerabilities identified in the security review?

## Test Your Entire Solution

Test the full scope of your solution using manual testing and automated security scanner tools. When you perform security scans, include all external endpoints that run independently of the Salesforce platform. Document false positive security violations and fix all code that doesn't meet Salesforce security guidelines.

### Testing Scope


Test all pieces of the solution that you submit for security review. Ensure that the solution architecture is secure, including endpoints that aren't hosted on the Salesforce platform. Your attention to all components and layers of your solution helps minimize the risk of hackers or malware exploiting potential entry points.

The full scope of your solution is subject to security review testing. For example, we can perform penetration tests that attack your Development Edition test org and attempt to access sensitive data or authenticate with false credentials.

To determine testing scope, use a follow-the-data approach. Wherever the customer or data goes is in scope. For example, your Salesforce customer is required to log in to your company website, or data is synced to a third-party server. Test these pieces to ensure that they're securely transferring credentials and data.


When either of the following criteria is true, external endpoints are within the scope of the security review and a required part of your security testing.

- The endpoint plays a role in authenticating the end user as part of buying, getting support for, or using your solution. This definition includes a connected app that doesn't require manual credential entry.
- Salesforce data is transferred to or from the endpoint.

 **Important:** Before you perform security testing on external endpoints that you don't own, complete two actions. First, obtain any necessary permission to perform security testing from the third parties that own the external endpoints. Second, follow the guidelines in [Salesforce IP Addresses & Domains to Allow](#).

### Automated Scanning Tools

To identify security vulnerabilities in your solution and external endpoints, we require that you run specific automated security scanning tools.

 **Tip:** We strongly recommend that you run security scans on your code and any connected endpoints throughout the development lifecycle. Run periodic scans and fix flagged issues as you go to prevent security vulnerabilities from piling up and creating more work for you later.

On the Partner Security Portal, you can access two Salesforce-supported security scanners: the Source Code Scanner, also referred to as the Checkmarx scanner, and the Chimera scanner. You can, and sometimes are required to, use scanners that aren't on the Partner Security Portal.

This table summarizes the automated security scanner tools that we require or recommend.

Security Scanner Tool	Scan Targets	Considerations	Results Accepted with Submission	Hosted on the Partner Security Portal
Source Code Scanner (Checkmarx)	Apex, Visualforce, and Lightning code	<ul style="list-style-type: none"> <li>• This static scanning tool uses Checkmarx security technology.</li> </ul>	Yes	Yes

Security Scanner Tool	Scan Targets	Considerations	Results Accepted with Submission	Hosted on the Partner Security Portal
		<ul style="list-style-type: none"> <li>You must provide a Checkmarx scan for any security review submission that includes a Salesforce package or component. These scans aren't required for mobile clients or API solutions.</li> <li>You're provisioned three Source Code Scanner runs per solution version with the security review fee.</li> <li>If you want the flexibility and freedom to scan unpackaged code, or bypass the three scan limit and package linking requirements, purchase a license from Checkmarx.</li> </ul>		
PMD Source Code Analyzer	Apex code	<ul style="list-style-type: none"> <li>The PMD scanner is a free, open-source tool.</li> <li>This tool is an alternative to the Source Code Scanner for solutions that contain Apex code.</li> <li>As you prepare your solution for security review, and as a supplement to the Source Code scanner, run PMD scans an unlimited number of times.</li> <li>PMD typically reports more false positives than Source Code Scanner tool.</li> </ul>	No	No
Salesforce CLI Scanner Plug-In	Apex, JavaScript, Lightning, TypeScript, and Visualforce code	<ul style="list-style-type: none"> <li>The Salesforce CLI Scanner plug-in unifies multiple static scanning tools, ESLint, JavaScript, PMD, and Retire JS, into one easy-to-install Salesforce CLI plug-in.</li> <li>You can install the CLI Scanner plug-in on a local development machine or integrate it into a continuous integration (CI) process.</li> <li>It includes customized rules to scan Lightning Web Component Javascript.</li> <li>It doesn't scan external endpoints.</li> <li>The CLI Scanner plug-in offers multiple output formats: csv, html, json, and junit.</li> </ul>	No	No
Chimera	External endpoints on domains that you own	<ul style="list-style-type: none"> <li>Chimera checks external endpoints of a solution.</li> <li>Chimera scans solutions from a Salesforce IP address.</li> <li>This scanner doesn't require a download.</li> <li>You can use Chimera with endpoints on domains that you don't own because it requires upload of a token to the root of the external server.</li> <li>If your solution connects to external endpoints that you don't own, use OWASP ZAP or Burp Suite.</li> </ul>	Yes	Yes

Security Scanner Tool	Scan Targets	Considerations	Results Accepted with Submission	Hosted on the Partner Security Portal
OWASP Zed Attack Proxy (ZAP)	External endpoints	<ul style="list-style-type: none"> <li>The ZAP Scanner is a free, community-driven proxy for web app security testing.</li> <li>Zap requires a download.</li> <li><a href="#">Setting Up ZAP for Browser</a> provides guidance for initiating security scans with this tool.</li> </ul>	Yes	No
Burp Suite	External endpoints	<ul style="list-style-type: none"> <li>Salesforce doesn't provision Burp Suite licenses for security review. Purchase a license independently.</li> <li>Burp Suite requires a download.</li> </ul>	Yes	No

## SEE ALSO:

[Security Scanners on the Portal](#)[False Positives](#)[PMD Source Code Analyzer Project Apex Rules](#)[OWASP Zed Attack Proxy \(ZAP\)](#)[Burp Suite](#)

## False Positives

---

As you navigate the AppExchange security review process, you're likely to encounter *false positive* issues with your solution. A false positive occurs when a security-scanning tool or code reviewer flags code that appears to pose a security vulnerability but actually doesn't. Instead, the flagged vulnerability is nonexistent, nonexploitable, or not required to support a valid use case or functionality.

Improve your likelihood of passing an initial or follow-up security review by addressing false positives in your submission. Include a document that explains why each flagged false positive doesn't pose a security risk.

### [Document Your Responses to False Positives](#)

Most often, false positives appear in Source Code Scanner (Checkmarx), Chimera, ZAP, or Burp Suite scanner results. False positives occasionally show up in Salesforce security review failure reports. In either case, you can improve your likelihood of passing security review by including a false-positive explanatory document when you submit your code.

### [Example Responses to False Positives in Checkmarx Scan Results](#)

The following example shows how to document your responses to false positives resulting from a Checkmarx scan. The example is in tabular format, but you can use whatever format suits the reporting of your information.

### [Example Responses to False Positives in a Security Review Failure Report](#)

The following example shows how to document your responses to false positives listed in a Salesforce security review failure report. It's written to support a retest submission.

## Document Your Responses to False Positives

Most often, false positives appear in Source Code Scanner (Checkmarx), Chimera, ZAP, or Burp Suite scanner results. False positives occasionally show up in Salesforce security review failure reports. In either case, you can improve your likelihood of passing security review by including a false-positive explanatory document when you submit your code.

You can use any format to document a false-positive response. For each flagged issue, include:

- Location—State the code location of the reported vulnerability.
- Explanation—Explain why the flagged code doesn't pose a vulnerability.

In addition to providing rationales for false positives, include in your documentation explanations that clarify special use cases, circumstances, or exceptions.

Some categories of security scan results are false positives that don't require documentation or code reworking. These categories exist in most of the security scanners that we accept for security review. Other scan results fall into severity categories that require attention because they highlight known security vulnerabilities. If you can't submit justifiable false positive documentation, rework the flagged code to meet security standards.

Scanner	Scan Results Requiring Attention for Security Review	Scan Results Not Requiring Attention
Source Code Scanner (Checkmarx)	All issues regardless of severity level that aren't labeled "Code Quality"	Issues labeled "Code Quality"
ZAP and Burp Suite	Issues categorized as high severity	Action on low and medium severity issues isn't required, but investigation into whether they pose a security threat is encouraged.
Chimera	All issues regardless of severity level that aren't labeled "Informational/Other"	Issues labeled "Informational/Other"

## Example Responses to False Positives in Checkmarx Scan Results

The following example shows how to document your responses to false positives resulting from a Checkmarx scan. The example is in tabular format, but you can use whatever format suits the reporting of your information.

Reported Vulnerability	Location	Response
FLS Update	Paths 1–17	We implemented and called the AuthManager class to check these paths for us or throw an error. You can see that in <code>ControllerFile.cls</code> on lines 241, 245, and 249.
FLS Update	Paths 18–24	Have been fixed and are valid.
FLS Update	Paths 25, 26, and 30	Are against our custom object UsageLog__c and not intended for user consumption. They are never exposed to users directly.
FLS Update	Paths 27–29	Must update the Account.NumberRelatedIssues__c field to

Reported Vulnerability	Location	Response
		appropriately count the new object created, irrespective of user input.
Sharing Violation	BatchCleanData.cls	We minimized the functions that this class calls to only the minimum set that requires <code>without sharing</code> .
Sharing Violation	LightningController.cls	Changed declaration to <code>with sharing</code> .
Sharing Violation	GlobalIssueReporting.cls	Changed to <code>useInherited sharing</code> because we don't know which context our calling class requires.
Stored XSS	Issue.page file: paths 1–3	<code>reportIssueList</code> is a list of <code>objectID + '' + integers</code> . It poses no XSS risk.
Stored XSS	Issue.page file: path 4	Fixed by removing <code>escape="false"</code> .
Stored XSS	Issue.page	We sanitized <code>usageLog</code> in JavaScript using the Salesforce <code>SecureFilters</code> library.

## Example Responses to False Positives in a Security Review Failure Report

The following example shows how to document your responses to false positives listed in a Salesforce security review failure report. It's written to support a retest submission.

Reported Vulnerability	Location	Response
Insecure Software Version	jQueries	Updated.
Insecure Software Version	moment.js	No user input flows into moment parsing. User input flows only to Salesforce Date fields.
Insecure Storage of Sensitive Data	UserConfig__c.object	The <code>apiKey__c</code> field is encrypted before setting with the encryption key, which is stored in a protected custom setting.
Insecure Storage of Sensitive Data	IssueInvite__c.object	The <code>password__c</code> field is a support-agent selected password to share resources publicly with the internet. It's not a user-owned secret.
Insecure Storage of Sensitive Data	APIManagement__c.object	We deprecated this custom setting, but it's impossible to delete custom setting definitions from managed packages.
Insecure Storage of Sensitive Data	AuthManager.cls	The credentials in comments are only example credentials. They do not authenticate to any development or production system.

Reported Vulnerability	Location	Response
Stored XSS	<a href="https://content.saslesforce.partner.com">https://content.saslesforce.partner.com</a>	We spoke to Jane Doe at Salesforce during office hours on Feb. 1, 2020. This URL is linked to a nonsensitive content domain. The URL has no session data to access back-end information. We were told that this finding could be a false positive.

## Security Review Resources

---

These resources can help you prepare for the AppExchange security review.

- [AppExchange Security Review](#) on page 133
- [Security Review Requirements Checklist](#)
- [Prevent Common Violations of Secure Coding Guidelines](#) on page 20
- [Secure Cloud Development Resources](#)
- [Secure Coding Guide](#)
- [Open Web Application Security Project \(OWASP\)](#)
- [OWASP Top 10 Issues](#)
- [OWASP Testing Guide](#)
- [OWASP Secure Coding Guide](#)
- [OWASP Secure Coding Practices Quick Reference](#)

## CHAPTER 7 Publish Your Solution on AppExchange

### In this chapter ...

- [What Is AppExchange?](#)
- [Business Plans for AppExchange Listings](#)
- [Submit a Due Diligence and Compliance Certification](#)
- [Publish on AppExchange](#)
- [How Does AppExchange Search Work?](#)
- [Email Notifications](#)
- [Collect AppExchange Leads](#)
- [Analytics Reports for Publishers](#)
- [Update the Package in Your 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 AppExchange?

AppExchange is the Salesforce marketplace, offering thousands of solutions and services that extend Salesforce. If you're an ISV partner or consultant, AppExchange helps customers discover your solution or service. If you're a Salesforce admin or user, AppExchange helps you find tools and talent to unleash your company's productivity.

## How Does AppExchange Work?

An AppExchange listing is your primary marketing tool for promoting your solution. 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're 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 create a provider profile and upload your solution package, you can create a listing. You can create only one listing per solution. 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 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 AppExchange?

Anyone can browse listings and test-drive solutions. You need the "Download Packages" permission to install solutions in your org. 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.  <b>Note:</b> We ask you to provide compliance information only for paid AppExchange listings.

Listings · My App Listing Preview Publish Listing

**1 BUSINESS PLAN** TEXT MEDIA APP TRIALS LEADS PRICING

Share your business plan with Salesforce to help us learn about your product. We'll review the plan and give you feedback on how to maximize your AppExchange success. For more information about business plans, see the [AppExchange Trailblazer Checklist](#).

**2 Business Details**

Tell us about your company and the market for this product.

**Complete**

- 1. Company Information
- 2. Business Plan
- 3. Product Information
- 4. Personas

**3 Product Architecture**

Tell us about the technical aspects of this product and how it integrates with Salesforce.

**In Progress**

- 1. User Experience
- 2. External Components
- 3. Security
- 4. Enterprise Scale
- 5. Salesforce Components

**4 Compliance Certification**

Verify that your company's business practices meet our standards for ethics and integrity.

**Not Started**

- 1. Contact Information
- 2. Business Information
- 3. Government Relationships
- 4. Principals and Key Employees
- 5. Compliance Information
- 6. Financial Information

**5 Salesforce Approval**

After you provide the required details, submit your business plan to Salesforce. We'll review the submission and contact you.

**Incomplete**

Your business plan is missing required details

Submit

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.

**6 AppExchange Partner Agreement** Type ISVforce Clickthrough

**Not Started**

Open Agreement

## Submit a Due Diligence and Compliance Certification

If you're starting a Salesforce consulting practice, submit a Due Diligence and Compliance Certification on behalf of your company. In the certification, you share details about your company and its business practices and relationships. We review these details to ensure that your company meets our standards for ethics and integrity.



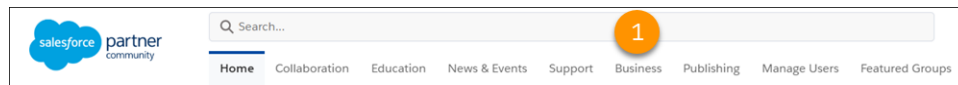
**Tip:** At many companies, filling out the certification is a team effort. To allow other people to edit your company's certification, assign the Manage Partnership permission.

1. Join the [Salesforce Partner Community](#).
2. If prompted, click **Go to Certification**. Otherwise, click the **Business** tab (1).

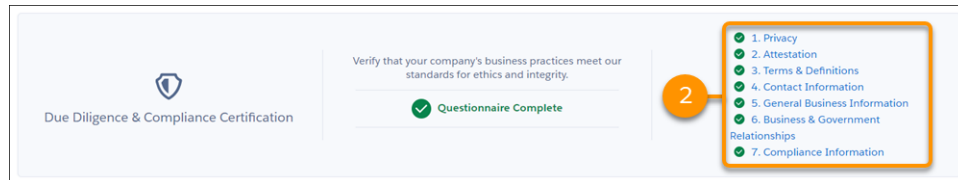
### USER PERMISSIONS

To edit the Due Diligence and Compliance Certification:

- **Manage Partnership**

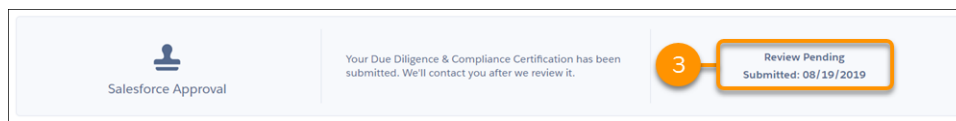


3. For Due Diligence & Compliance Certification, click a questionnaire section (2) to provide the related details.



4. After you've provided all the required information, submit your certification for review.

To check the status of the review (3), go to the Salesforce Approval section. If we approve your certification, you're a step closer to launching your Salesforce consulting practice. Approved practices get access to tools and features for consulting partners, such as the ability to log completed implementation projects.



## Publish on AppExchange

Learn how to publish your solution or consulting service listing on AppExchange.

### [Connect a Packaging Org to the Publishing Console](#)

To add a package to an AppExchange solution listing, first connect the packaging org associated with that package to the Publishing Console.

### [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.

### [Create or Edit Your AppExchange Listing](#)

Market your solution or consulting service by listing it on AppExchange. Create or edit a listing that stands out to customers.

### [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.

### [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.

### Select an Installation Option

The easier it is for people to install your solution, the more likely it is they become paying customers. Offer the option that gives your customers the best installation experience.

### 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.

### Complete the Security Review Cycle

To distribute a solution on AppExchange, it must pass our comprehensive security review. Use the security review submission interface in the Salesforce Partner Community to manage your reviews. Submit your solution for an initial review. Resubmit a solution you revised to correct security issues detected in a previous review. Pay security review and AppExchange listing fees.

## Connect a Packaging Org to the Publishing Console

To add a package to an AppExchange solution listing, first connect the packaging org associated with that package to the Publishing Console.

1. Log in to the [Salesforce Partner Community](#).
2. Click the **Publishing** tab.
3. Click the **Organizations** tab.
4. Click **Connect Org**.
5. Enter the login credentials for the org that contains the package you want to list.
6. Click **Submit**.

If any packages are found in the connected org, they appear on the **Packages** tab of the Publishing Console. From the **Packages** tab, you can create a listing for your packaged solution, or submit the solution for 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 Your AppExchange Listing

Market your solution or consulting service by listing it on AppExchange. Create or edit a listing that stands out to customers.

Open the Publishing Console in the Salesforce Partner Community, and click the **Listings** tab. To create your listing, click **New Listing**. To edit your listing, find and open the listing.

Here are the tabs you navigate when creating or editing your listing.

Tab	What you do:	Available on these listings types:
Business Plan	<ul style="list-style-type: none"> <li>• Add a business plan for your offering.</li> <li>• If you're a standard AppExchange partner, sign your Salesforce partnership agreement.</li> </ul>	All solution types

Tab	What you do:	Available on these listings types:
Service Offering	<ul style="list-style-type: none"> <li>Choose listing categories, such as services offered and industry focus.</li> </ul>	Consulting services
App, Component, Data Set, Flow, or Bolt	<ul style="list-style-type: none"> <li>Upload the package that contains your solution, or provide an installation link.</li> </ul>	App, Component, Lightning Data, Salesforce Flow, or Lightning Bolt solutions, respectively
Security Review	<ul style="list-style-type: none"> <li>Submit a solution for security review.</li> <li>Check the status of a review.</li> <li>View a listing's review history.</li> </ul>	All solution types
Text	<ul style="list-style-type: none"> <li>Describe your offering.</li> <li>Provide contact information so that customers and Salesforce can get in touch.</li> </ul>	All solution types and consulting services
Media	<ul style="list-style-type: none"> <li>Add branding.</li> <li>Upload images, videos, and other resources to help customers understand your offering.</li> </ul>	All solution types and consulting services
Trials	<ul style="list-style-type: none"> <li>Set up a test drive or free trial so that customers can see your offering in action.</li> </ul>	All solution types
Leads	<ul style="list-style-type: none"> <li>Choose how Salesforce collects leads when customers interact with the listing.</li> </ul>	All solution types
Pricing	<ul style="list-style-type: none"> <li>Choose whether your offering is free or paid and provide pricing information.</li> </ul>	All solution types

## 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.

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.

### USER PERMISSIONS

To edit AppExchange listings:

- Manage Listings

 **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](#).

## Select an Installation Option

The easier it is for people to install your solution, the more likely it is they become paying customers. Offer the option that gives your customers the best installation experience.

Option	When to choose this option:
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.

Option	When to choose this option:
	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 <b>Get It Now</b> 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**.

## Complete the Security Review Cycle

To distribute a solution on AppExchange, it must pass our comprehensive security review. Use the security review submission interface in the Salesforce Partner Community to manage your reviews. Submit your solution for an initial review. Resubmit a solution you revised to correct security issues detected in a previous review. Pay security review and AppExchange listing fees.

### [Required Materials for Security Review Submission](#)

Learn about the materials that you must provide, such as test environments and documentation, when submitting your solution for an AppExchange security review. Mobile apps have platform-specific submission requirements. Extension packages undergo security review and Salesforce requires the same materials for them as for a standalone solution.

### [Security Review and AppExchange Listing Fees](#)

When you submit your solution for an initial security review, you pay security review and AppExchange listing fees. We waive the fees for solutions that are distributed for free on AppExchange. If we find security vulnerabilities in your solution, you can fix and resubmit it a limited number of times at no extra charge.

### [Update Your Payment Information](#)

Update the payment information that's on file for your AppExchange listings from the Security Review tab of your listing.

### [Submit Your Solution for Security Review](#)

Use the security review submission interface in the Salesforce Partner Community to submit your solution for review. Share your solution and all required materials, and pay applicable fees.

### [Check Security Review Progress and History](#)

After you submit your solution for review, find key review information in the progress and history components on your listing's Security Review tab. Check the progress component for review status. Scan the history component for data on past and current reviews.

### [Act on Security Review Results](#)

Approximately 4–6 weeks after you submit a solution for an initial review, your security review report arrives in your inbox. Check the report to learn if your solution is or isn't approved. Learn how to request a follow-up review for a solution that isn't approved and how to publicly list an approved solution.

### [Periodic Re-Reviews](#)

We conduct periodic re-reviews for all solutions listed on AppExchange. Re-reviews ensure that solutions continue to safeguard against the latest security vulnerabilities.

## Required Materials for Security Review Submission

Learn about the materials that you must provide, such as test environments and documentation, when submitting your solution for an AppExchange security review. Mobile apps have platform-specific submission requirements. Extension packages undergo security review and Salesforce requires the same materials for them as for a standalone solution.



During a security review, Product Security tests the required and optional parts of your solution. To determine testing scope, we typically use a follow-the-data approach. Wherever the customer goes, we go. For example, to use your solution, your Salesforce customer needs an account on your company website, or data is synced to a third-party server. Our review team tests these pieces to ensure that they're securely transferring Salesforce credentials and data.

Provide access to all environments, packages, and external components that your solution uses, including:

- External web applications or services.
- Client or mobile applications that are required or optional.
- All Apex and Visualforce that is included in your solution.

 **Note:** Be sure that your submission is a Managed—Released package. We can't accept an unmanaged or beta package.

If you're not sure whether to include part of your solution, include it anyway. The review team doesn't test parts that are out of scope, but omitting a required part delays your review.

We like to see that you did your due diligence to ensure that your solution meets enterprise security standards. Include security scan reports along with explanations of any false positives that appear in your test results.

We also ask for detailed solution user documentation and your company's information security policies. We understand that providing extensive documentation isn't practicable for smaller or newer companies, so we factor in company size and maturity when reviewing submitted documents.

To generate a checklist that is customized to your solution, use the [Security Review Submission Requirements Checklist Builder](#) in the Salesforce Partner Community. Here's the checklist for a Lightning Component.

### Security Review Submission Requirements Checklist Builder

#### Tell Us About Your App

**Select all details below that apply to your app in order to create a custom checklist of required information in your Security Review submission**

On Salesforce Platform

Apex/Visualforce Package

Lightning Component

Quip App

Marketing Cloud App

External to Salesforce Platform

Website

API Endpoints

Mobile App

Browser Extension

Desktop/Client App

Compile Checklist

#### Your Checklist

- 1

**Force.com Source Code (Checkmarx) Scanner Results**

  - Access the scanner via the [Partner Security Portal](#) >
  - Please resolve any issues ahead of the scan to show clean results
  - For any issues on the report which can't be resolved, be sure to explain in a separate false positive document
- 2

**Salesforce test environment:**

  - Create a Developer Edition org via [Environment Hub](#) or [here](#)
  - Install your managed package in the org
  - Populate test data
  - Provide credentials for an admin-level user
  - [Enable My Domain](#) if package contains Lightning >
- 3

**Basic app usage instructions to help orient the Security review team**

\* Prior to submitting, steps 1-8 on the [Trailblazer Checklist](#) must be completed.

The following table summarizes what to submit based on the scope of your architecture.

Material for Submission	Salesforce Native Solution	Salesforce Native Solution with Lightning Components	Solution with External Web App or Service	Solution with a Mobile Client	API Only	Marketing Cloud App
Salesforce Developer Edition org	X	X	X	X	X	
Managed package installed a in Developer Edition org	X	X	X	X		
URLs and login credentials for external components requiring authentication			X	X	X	
Checkmarx report	X	X	X	X		
Zap or Chimera scan report			X	X	X	X
False positives documentation (if applicable)	X	X	X	X	X	X
Solution documentation	X	X	X	X	X	
Platform with installation link or file				X		
Credentials to Marketing Cloud environment						X

## Mobile Apps

For mobile app testing, provision the app for all the platforms that you plan to distribute on. For iOS, we accept a test flight or an ad hoc deployment. For other platforms, we accept the app in a file, such as an Android Packaging (.apk) file.

## Extension Packages

An extension package is a package that is an add-on to a solution or that integrates the functionality of two solutions. Before you can publicly list an extension package on AppExchange, it and the solutions it extends must pass security review.

If your extension package is an add-on to, or integrates with, base solutions that *have passed* the security review, submit only your extension package for review. However, if the base solutions *haven't passed* the security review, submit your extension package plus the unreviewed solutions.

The security review submission requirements for an extension package are the same as for a solution that has a similar architecture. For example, if you have an extension package with external callouts, attach separate web scan results for the packages with the callouts.

The Product Security team reviews the solution as a whole. Install a complete solution in the Development Edition org that you submit with your security review. Include your extension package. Also install all base and dependent packages for the solutions that your package extends or integrates. That's required whether the base solutions have already passed the security review or not.

It's important that the Salesforce security team reviews every extension package. Even small packages can introduce security vulnerabilities.

#### SEE ALSO:

[False Positives](#)

[Security Review Requirements Checklist Builder](#)

## Security Review and AppExchange Listing Fees

When you submit your solution for an initial security review, you pay security review and AppExchange listing fees. We waive the fees for solutions that are distributed for free on AppExchange. If we find security vulnerabilities in your solution, you can fix and resubmit it a limited number of times at no extra charge.

Expect to pay a one-time, upfront total of \$2,700 USD to initiate the security review process for most solutions. This payment includes a \$2,550 security review fee and \$150 first-year AppExchange listing fee. Also expect to pay the \$150 AppExchange listing fee annually after the first year. The annual listing fee includes reviews for updates to solutions already listed on AppExchange.

Fees are refundable. Refund requests must be made within 180 days of payment.

If you have questions about security review or listing fees, contact your Partner Account Manager or log a support case in the Salesforce Partner Community.

Here's a detailed breakdown of the fees.

Review Type	Security Review Fee	Annual Listing Fee
Initial review of a paid solution	\$2,550 USD	<ul style="list-style-type: none"> <li>\$150—first-year fee is added to the security review fee and due upon submission</li> <li>\$150 charged annually after the first year of listing</li> </ul>
All reviews of free solutions	No charge	<ul style="list-style-type: none"> <li>No charge</li> </ul>
Follow-up review of a paid solution resubmitted after addressing issues detected in a previous security review	No extra fee—a limited number of follow-up reviews are included in the security review fee paid at original submission of solution	<ul style="list-style-type: none"> <li>First-year listing fee paid at initial review submission</li> <li>\$150 fee charged annually after first year of listing</li> </ul>
Periodic re-review of a previously approved, listed, paid solution	No extra fee—cost included in the security review fee paid at original submission of solution	<ul style="list-style-type: none"> <li>First-year listing fee paid at initial review submission</li> </ul>

Review Type	Security Review Fee	Annual Listing Fee
		<ul style="list-style-type: none"> <li>\$150 fee charged annually after first year of listing</li> </ul>

When you submit your solution for an initial review, be prepared to pay the review and first-year listing fees. There are no fees for solutions that you distribute for free on AppExchange.


When you submit a solution for a follow-up or re-review, you're always prompted to confirm your credit card information. However, you're not always charged.

Often, the initial security review and annual listing fees include follow-up and periodic reviews. For example:

- **You resubmit a solution that fixes security issues discovered in an initial review.** Your resubmission exclusively fixes issues discovered in an initial security review, and you fix the code in the existing package. We conduct a follow-up review. If you make other revisions, such as functionality changes, we require that the revised solution go through an initial security review. That's also true if you spin up a new package for the revised code.
- **You list a new version of a package that we previously approved.** You upload the new version to AppExchange and associate it with your listing. To identify security vulnerabilities, we run a source code scan. This scan is included in the annual listing fee.
- **You create a managed package to upgrade your offering.** You develop the new version in a package that we previously approved. The upgrade is automatically approved when you submit it for review. You can immediately associate the new version to your listing. We review the new version 6 months to 2 years after the solution is listed, depending on potential risk of the solution. The review is included in the initial security review fee.

## Update Your Payment Information

Update the payment information that's on file for your AppExchange listings from the Security Review tab of your listing.

 **Note:** Your listing must have an active subscription.

1. Log in to the [Salesforce Partner Community](#).
2. Click the **Publishing** tab.
3. Open any listing that you submitted for security review.
4. Click the **Security Review** tab.
5. Click **Update Payment**.
6. Edit your payment information.
7. Click **Update**.

### USER PERMISSIONS

To access the Salesforce Partner Community Publishing Console:

- Manage Listings

## Submit Your Solution for Security Review

Use the security review submission interface in the Salesforce Partner Community to submit your solution for review. Share your solution and all required materials, and pay applicable fees.

Before you submit your solution for security review:

- Receive Salesforce approval of your business plan.
- Have a partner recruitment representative confirm that your solution is enrolled in the AppExchange Partner Program, and that you have a distribution agreement.

### USER PERMISSIONS

To access the Salesforce Partner Community Publishing Console:

- Manage Listings

- Create your AppExchange solution listing in the Partner Community Publishing Console. If your solution includes a package, [connect your packaging org to the Publishing Console](#) and [upload the package](#).
  - Configure a Developer Edition test environment with your solution installed. We use the environment to test your solution.
  - Certify that your solution is Lightning Ready. All new solutions submitted for security review must be Lightning Ready.
1. Log in to the [Salesforce Partner Community](#).
  2. Navigate to the Business Plan tab of your solution listing in the Publishing Console.
  3. Verify that your business plan is approved.
  4. If your plan is approved, click the solution-type tab. The tab name corresponds to the type of solution you're listing: App, Component, Flow, and so on.
  5. Provide solution details.
  6. If your solution includes a package, click **Select Package**, then find and select the managed package version that you plan to list. If you can't find the package to associate with your listing, check that the packaging org is connected to the Publishing Console.
  7. Save your changes.
  8. On the Security Review tab, click **Start Review**. You're guided through the options and settings that require your input, based on the solution type.
  9. On the Payment page, select an option for Pricing. Your selection must match the selection on the listing's Pricing tab.
    - a. If you plan to sell your solution on AppExchange, select **Paid** and enter payment information. Solutions that require payment later, such as free downloads, free installs, and pay-per-use are considered paid.
    - b. If you plan to distribute your solution for free, select **Free**. We don't charge security review or annual listing fees for solutions that are distributed for free on AppExchange.
  10. Click **Submit**.

If anything is missing from your submission, the security review team contacts you. When everything is in place, they send you an email confirming that your solution is queued for review. The initial security review of your solution lasts 4–6 weeks. You can expect to receive your report soon after the review is completed.



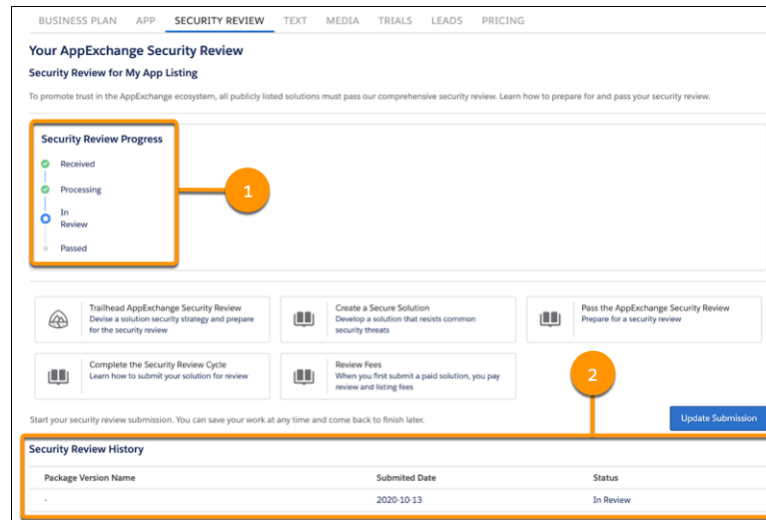
**Tip:** Schedule a technical office hours appointment right when you receive your confirmation email. Visit the [Partner Security Portal](#) and choose a date 4–6 weeks away. If your solution doesn't pass, you have an appointment booked.

## Check Security Review Progress and History

After you submit your solution for review, find key review information in the progress and history components on your listing's Security Review tab. Check the progress component for review status. Scan the history component for data on past and current reviews.

The Security Review Progress component (1) gives you a quick look at the current review's status, tasks that require your attention, and requests from the Product Security team.

The Security Review History component (2) catalogs security review activity for a specific listing. For each submitted review, past and present, see the package version name, submission date, and status.



With all this information in one place, getting post-submission updates has never been simpler.

Status	Component	Description
Received	Progress and History	Security Review Operations received your submission.
Processing	Progress and History	Security Review Operations is verifying that your submission is ready to review. A submission is ready to review if it includes everything required to test the security of your solution.
In Review	Progress and History	The review is in progress.
Passed	Progress and History	Product Security finished the review, and your solution passed. You can publicly list and distribute your solution on AppExchange.
Expired	History	<p>Either we received no response to re-review notifications and warnings, or the solution repeatedly failed re-review.</p> <p>To ensure that you receive re-review notifications, keep the contact information on your listing up to date.</p> <p>If you need more time to fix issues discovered in a re-review, request a deadline extension by logging a case in the Salesforce Partner Community. For the case topic,</p>

Status	Component	Description
		select Security Review under AppExchange (ISV) App Setup & Management.

SEE ALSO:

[Required Materials for Security Review Submission](#)

## Act on Security Review Results

Approximately 4–6 weeks after you submit a solution for an initial review, your security review report arrives in your inbox. Check the report to learn if your solution is or isn't approved. Learn how to request a follow-up review for a solution that isn't approved and how to publicly list an approved solution.

### [Submit Your Solution for a Follow-Up Review](#)

The security review of your solution is complete, but the Product Security team found security vulnerabilities. Your solution isn't approved for distribution on AppExchange. It's not the result you hoped for, but you're in good company. Most solutions don't pass on the first try. Fix the vulnerabilities, and submit your solution for a follow-up review.

### [List Your Solution on AppExchange](#)

Your security review is complete and your solution passed. Congratulations! Publicly list and distribute your solution on AppExchange.

## Submit Your Solution for a Follow-Up Review

The security review of your solution is complete, but the Product Security team found security vulnerabilities. Your solution isn't approved for distribution on AppExchange. It's not the result you hoped for, but you're in good company. Most solutions don't pass on the first try. Fix the vulnerabilities, and submit your solution for a follow-up review.


A security review report for a solution that didn't pass lists the types of security vulnerabilities that Product Security found. For each vulnerability type, the report includes:

- A specific example from your solution.
- Steps to reproduce the issue.
- Links to documentation or comments about how to fix the issue.

Our goal is to find as many different types of vulnerabilities as possible, but keep in mind that the security review is a black-box, time-limited process. We can't always list every instance of a security vulnerability, and we may not initially detect all issue types. Interpret the security review findings as representative examples of the types of issues you must fix. Unless otherwise noted in the report, you're required to fix all classes of issues across the entire solution.

We're available to help you analyze the findings and troubleshoot security vulnerabilities. Schedule a technical office hours appointment on the [Partner Security Portal](#).

As you revise your solution, exclusively fix security issues discovered in a previous review, and fix the code in the existing package. If you make other revisions, such as functionality changes, we require that the revised solution go through an initial security review. That's also true if you spin up a new package for the revised code.

 **Important:** If the package ID and namespace don't change, your resubmission qualifies for a follow-up review.

After you fix the solution, collect the materials necessary for us to complete a follow-up review. Rerun the required scanner tools on your revised solution and generate updated scan reports. If you fixed issues in your managed package, provide updated Source Scanner

results. If you fixed issues detected on an external endpoint, provide updated ZAP or Chimera scan reports. If applicable, [document your responses to false positives](#) on page 142.

For more details about what to submit, see [Required Materials for Security Review Submission](#) on page 153.

The process to request a follow-up review depends on the scope of changes.

- **New Package Version:** You fixed code that runs on the Salesforce platform. Create and upload a new version of your managed package to your AppExchange listing. Then start a review for the new version. If you also made changes external to the package, include details with your submission.
- **External Code or API-Only Solution:** You only changed code that runs externally to Salesforce. Edit your existing security review submission. Provide details about the changes. Log a security review case so that Product Security knows you're resubmitting your solution.

#### [Request a Follow-Up Review for a New Package Version](#)

To fix security vulnerabilities discovered in a previous review, you changed code that runs on the Salesforce platform. Upload a new version of your managed package, associate it with your AppExchange listing, and request a follow-up review. If you also made changes external to the package, include details with your security review submission.

#### [Request a Follow-Up Review for External Code or an API-Only Solution](#)

To fix security issues discovered in a previous review, you changed only the code that runs externally to Salesforce or you changed an API-only solution. To request a follow-up review, edit your existing security review submission and log a case in the Salesforce Partner Community.

#### SEE ALSO:

[Document Your Responses to False Positives](#)

[Required Materials for Security Review Submission](#)

### Request a Follow-Up Review for a New Package Version

To fix security vulnerabilities discovered in a previous review, you changed code that runs on the Salesforce platform. Upload a new version of your managed package, associate it with your AppExchange listing, and request a follow-up review. If you also made changes external to the package, include details with your security review submission.

1. Upload the new version of your managed package to the Publishing Console.
2. Log in to the [Salesforce Partner Community](#).
3. Click the **Publishing** tab.
4. Click your solution's tile.
5. Click the solution-type tab. The tab name corresponds to the type of solution you're listing: App, Component, Flow, and so on.
6. Click **Select Package**, then find and select the new managed package version.
7. Update solution details as needed, then save your changes.
8. Click **Submitted**. You're guided through the options and settings that require your input. Update information as needed.
9. On the Payment page, if you're resubmitting a paid solution, review your stored payment information. A limited number of follow-up reviews are included in the security review fee paid with your initial submission. If you're within the limit, you aren't charged for the follow-up review. If you have questions about security review fees, contact your Partner Account Manager or log a support case in the Salesforce Partner Community.
10. Click **Submit**.

#### USER PERMISSIONS

To access the Salesforce Partner Community Publishing Console:

- **Manage Listings**

11. Log a security review case in the Salesforce Partner Community. Include your package name, ID, and version in the comments.

Logging a case lets Product Security know that you're resubmitting your solution.

If anything is missing from your submission, the security review team contacts you. When everything is in place, we send you an email confirming that your solution is queued for review. A follow-up review lasts 2–3 weeks. You can expect to receive your report soon after the review is completed.

## Request a Follow-Up Review for External Code or an API-Only Solution

To fix security issues discovered in a previous review, you changed only the code that runs externally to Salesforce or you changed an API-only solution. To request a follow-up review, edit your existing security review submission and log a case in the Salesforce Partner Community.

1. If your solution includes a package, upload the new version of your managed package to the Publishing Console.
2. Log in to the [Salesforce Partner Community](#).
3. Click the **Publishing** tab.
4. To request a follow-up review for external code:
  - a. Click the **Packages** tab.
  - b. Find the solution version that you want to submit.
5. To request a follow-up review for an API-only solution:
  - a. On the Listings tab, find and open your listing.
  - b. Click the solution-type tab. The tab name corresponds to the type of solution you're listing: App, Component, Flow, and so on.
6. Click **Submitted**.
7. If you're prompted to continue, click **Next**. You're guided through the options and settings that require your input. Update information as needed.
8. On the payment page, if you're resubmitting a paid solution, review your stored payment information. A limited number of follow-up reviews are included in the security review fee paid with your initial submission. If you're within the limit, you aren't charged for the follow-up review. If you have questions about reviews limits and security review fees, contact your Partner Account Manager or log a support case in the Salesforce Partner Community.
9. Click **Submit**.
10. [Log a security review case in the Salesforce Partner Community](#). Include your package name, ID, and version in the comments. Logging a case lets Product Security know that you're resubmitting your solution.

If anything is missing from your submission, the security review team contacts you on the open case. When everything is in place, we send you an email confirming that your solution is queued for review. A follow-up review lasts 2–3 weeks. You can expect to receive your report soon after the review is completed.

### USER PERMISSIONS

To access the Salesforce Partner Community Publishing Console:

- Manage Listings

## List Your Solution on AppExchange

Your security review is complete and your solution passed. Congratulations! Publicly list and distribute your solution on AppExchange.

1. Log in to the [Salesforce Partner Community](#).
2. Click **Publishing**.
3. Click **Listings**.

### USER PERMISSIONS

To access the Publishing Console:

- Manage Listings

4. Click your solution's tile and edit your listing as needed.
5. Click **Publish Listing**.
6. To confirm, click **Publish Listing** again.

Salesforce validates that your listing is ready to publish. For example, we check that you uploaded a tile image and that your solution passed the security review. After successful validation, your listing is published and visible to anyone visiting AppExchange.

SEE ALSO:

[How to Build a Perfect AppExchange Listing](#)

[Create or Edit Your AppExchange Listing](#)

## Periodic Re-Reviews

We conduct periodic re-reviews for all solutions listed on AppExchange. Re-reviews ensure that solutions continue to safeguard against the latest security vulnerabilities.

When you upgrade a managed package version of a solution that passed security review, you don't have to go through the full review process again. Submit the upgrade for review and it's automatically approved. You can immediately associate the new version to your AppExchange listing.

The automated review isn't the only security review of your upgraded solution. 6 months to 2 years after the solution is listed, we review the new version. This periodic re-review includes automated and manual tests. The actual timing depends on the potential risk of the solution.

To determine which listed solutions are due for re-review, we run risk-factor reports. If your solution shows significant change, it's likely that we conduct a re-review. When the time comes, we contact you to make arrangements. We also reserve the right to conduct random security penetration tests on your solution throughout the year.

There's no additional cost for re-reviews. These reviews are included in the security review fee paid at original submission of your solution.

If we find that your solution no longer meets our enterprise security standards, we notify you and provide a timeline to remedy the issues. In extreme cases, we pull the AppExchange listing from public viewing. Before you can relist it for distribution, you must fix the security issues and submit it for a follow-up review.

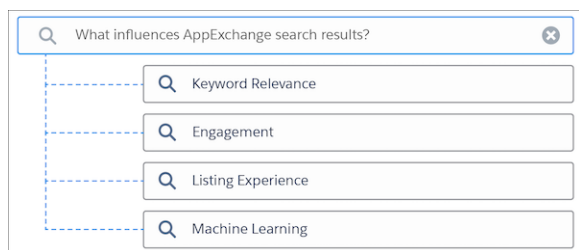
## How Does AppExchange Search Work?

---

Search is one of the most popular ways that Salesforce customers find solutions on AppExchange. Learn how keyword relevance, engagement, listing experience, and machine learning influence the search results that customers see. Then apply tips to help customers discover your listing when they search for a solution to a business problem.

## What Influences AppExchange Search Results?

When someone searches AppExchange, four factors influence the results they see. Keyword relevance is the most important factor, followed by engagement, listing experience, and machine learning.



## Keyword Relevance

Keyword relevance considers how closely customers' search terms align with text on your listing. The more that the search terms align with your listing text, the higher its keyword relevance. Title, tagline, and brief description text are weighted more heavily than other listing text.

### Example

A customer visits AppExchange to find an app for administering surveys. Their search includes the words feedback and collection. AppExchange listings that include these words have a higher keyword relevance than listings that don't.

## Engagement

Engagement is informed by your listing's popularity and considers customer activities like screenshot views, test drives, and installs. We measure these activities daily and in aggregate over the past 30 days. The more customer activities that occur on your listing, the higher its engagement.

### Example

A customer visits AppExchange to find a document generation app. After performing a search, they visit two listings. The first listing has only a few low-resolution screenshots, so the customer leaves without interacting. The second listing has high-resolution screenshots, a video, and a free trial, and the customer interacts with each of them. In this scenario, the customer's behavior contributes to higher engagement for the second listing than the first.

## Listing Experience

Listing experience considers other aspects of your listing that aren't included in keyword relevance and engagement factors. Some of these aspects relate to your Salesforce partnership, such as participation in the Pledge 1% program. Others relate to customers' experiences with your solution, such as the number and quality of reviews on your listing or when your solution was last updated.

### Example

A Salesforce partner lists a new telephony app on AppExchange. To promote awareness and installs, the partner launches a marketing campaign. Then the partner sends follow-up emails to customers who installed the app. The email thanks customers for trying the app and asks them to share their feedback on AppExchange. The number of reviews grows and listing experience increases.

## Machine Learning

Machine learning uses AI to improve the search experience on AppExchange. Like other search providers, we don't share details about our machine learning algorithm. But trust and customer success are central to the design of the algorithm. Trust means that the algorithm continuously tunes search results to ensure authenticity. Customer success means that the algorithm makes inferences about a customer's search intent and prioritizes the results that are most likely to drive positive outcomes.

**Example**

A customer visits AppExchange and searches for a solution called Appy's Maps. In the search results, a competing solution appears alongside Appy's Maps. This solution appears because some who searched for Appy's Maps eventually installed the competing solution. The machine learning algorithm considers this outcome positive and associates the competing solution with Appy's Maps.

## How Can I Make My Listing Easy to Find When Customers Search AppExchange?

Here are some tips to help your listing stand out in the AppExchange search results.

Factor	Tips
Keyword Relevance	<ul style="list-style-type: none"> <li>Identify the business problems that your offering solves, then select keywords for your listing. When you incorporate keywords into your listing, focus on the title, tagline, and brief description.</li> <li>Avoid keyword stuffing. If you pack your listing with too many or unrelated keywords, it's difficult for customers to understand the value it provides. Plus, it negatively affects the machine learning algorithms.</li> <li>Review the keywords that drive your listing activity by using Marketplace Analytics visualizations. These visualizations help you determine the keywords that are associated with the highest number of tile, video, and demo views. To gauge engagement, regularly review your analytics and improve your offering.</li> </ul>
Engagement	<ul style="list-style-type: none"> <li>Be sure that your listing features screenshots, graphic tiles, a video, and a demo. Engagement is enhanced when customers interact with your media, so focus on quality, not quantity.</li> <li>Entice your customers to scroll through screenshots that describe your solution's benefits. Add a call to action or a visual aid that directs customers to watch your demo video.</li> <li>Use your video to advertise your solution.</li> <li>Use your demo video to provide an in-depth look at your solution's features.</li> </ul>
Listing Experience	<ul style="list-style-type: none"> <li>Monitor the feedback that your solution receives. Respond to positive feedback with a thank you, and respond to negative feedback with helpful tips and solutions. AppExchange doesn't edit published reviews, but your customers can edit them based on their positive interactions with you.</li> <li>Keep your listing fresh. When you upload a new package and release a new version, review your listing content. Make sure to describe your current features and use the best-fit keywords.</li> </ul>

Factor	Tips
	<ul style="list-style-type: none"> <li>Keep up to date with Salesforce releases. Check that your solution works with our latest technology and update your listing accordingly.</li> </ul>

Maintaining a strong search position is a marathon, not a sprint. All search factors work together, and can change over time. Periodically review your listing's keywords, content, and analytics so that they contribute to machine learning. Make updates to those factors that you control.

#### SEE ALSO:

[Make Your AppExchange Listing Effective](#)

[Collect AppExchange Leads](#)

[How Do Customers Find My Listing?](#)

## 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.

Type of Email Notification	Sent to	Details
New Comment on the Review of Your Listing	You, the provider	Sent whenever someone writes a new comment on a review of your listing.

## Collect AppExchange Leads

You can configure your AppExchange listings to collect leads and deliver them to your Salesforce org. Specific customer interactions, such as watching your listing's demo video, can trigger lead collection.

SEE ALSO:

[What's the Difference Between Lead Events and Leads in Marketplace Analytics?](#)

[Generate Leads from Your Website for Your Sales Teams](#)

## AppExchange Leads

When you enable lead collection for your AppExchange listing and a customer interacts with the listing, AppExchange records a lead. If you enabled Web-to-Lead in your Salesforce org, AppExchange can also deliver the lead to that org. Some Web-to-Lead settings can prevent leads from being delivered to your org.

You can collect leads when a customer:

- Installs your solution
- Takes a test drive
- Watches a demo or video
- Signs up for a free trial
- Clicks **Learn More**

Before you enable lead collection on your listings:

- Configure Web-to-Lead in the org where you want to receive leads.
- Disable Require reCaptcha verification in the org's Web-to-Lead settings. If reCaptcha is enabled, no AppExchange leads are sent to the org.

Set up lead collection on a per-listing basis. For each listing, enable the customer interactions that trigger lead collection. For each interaction, also complete any required setup. For example, to collect leads when customers watch your demo, you must add a demo video to your listing.

When a customer interacts with your listing and lead collection is enabled for that interaction, they're prompted to fill out the AppExchange lead sign-up form. Info collected from the form, combined with customer activity data, is shared as a lead.



**Note:** You can't modify the lead form that customers are asked to fill out. To share ideas for improving the lead form, go to [IdeaExchange](#).

Regardless of your listing's lead-collection settings, customers can still view your demo, take a test drive, click to learn more, and install your solution.

## AppExchange Leads and License Activities

When you enable lead collection for your AppExchange listing and a customer interacts with your listing, AppExchange records a lead. License records are generated when a customer installs your solution.

AppExchange can generate leads when a customer takes the specific actions on the listing for which you chose to generate leads.

Leads can be generated when a customer:

- Installs your solution
- Takes a test drive
- Watches a demo or video
- Signs up for a free trial
- Clicks **Learn More**

By contrast, license records are generated only when a customer installs your solution. To receive licenses, you must also have the [License Management Application \(LMA\)](#) enabled in your partner business org.

## Enable AppExchange Lead Collection

Collect leads when customers interact with your AppExchange listings.

Before you enable lead collection, verify that the Salesforce org that receives leads is ready.

- You must receive leads in a standard Salesforce org, not a Developer Edition org.
  - The org where you receive leads must have Web-to-Lead enabled.
  - Require reCaptcha Verification must be disabled in your Web-to-Lead settings.
1. Log in to the [Salesforce Partner Community](#).
  2. Click the **Publishing** tab.
  3. On the Listing tab, click a listing tile.
  4. Click the **Leads** tab.
  5. Enable **Collect leads when customers interact with this listing**.
  6. Specify the Salesforce org where you want to receive the leads. We recommend using your partner business org so that you can manage leads and licenses from a single, convenient location.
  7. Enable lead collection for one or more activities.
  8. Save your changes.

### USER PERMISSIONS

To edit AppExchange listings:



- **Manage Listings**

## AppExchange Lead Source Codes

Lead source codes provide information about how the lead was created and can help you determine how to proceed.


AppExchange lead source codes always use this format: `SFDC-XX|Listing Name` or `SFDC-dup-XX|Listing Name.XX` identifies the action that the user performed to generate the lead.

This table lists AppExchange actions and what they mean.

Action	Description
IN	<p>The user clicked <b>Get It Now</b> on your listing and started the install process for your solution. This action includes agreeing to the terms and conditions and clicking the install button on the confirmation page.</p> <p> <b>Note:</b> Sometimes users don't complete the installation, or they uninstall your solution later. To track package installations, use the <a href="#">License Management App (LMA)</a>.</p>
DM	The user clicked <b>View Demo</b> on your listing and watched some or all of your demo video.
LM	<p>The user clicked <b>Learn More</b> on your listing.</p> <p> <b>Note:</b> Listings that previously had Learn More buttons now have Get It Now buttons and receive lead source codes with IN actions.</p>
TS	The user clicked <b>Get It Now</b> on your listing and started a 30-day free trial of Salesforce and your solution. These users can be existing Salesforce customers.
TD	The user clicked <b>Test Drive</b> on your listing and tried your solution in a test org.

## Package Installation Leads

Package installation is one example of a user activity that triggers lead creation. However, AppExchange isn't the only source of installation leads. The License Management App (LMA) also creates installation leads. Let's look at an example. A user purchases your solution and installs it via an installation URL. AppExchange isn't aware of the user's activity, so it doesn't create a lead. However, the installation triggers the LMA to create a lead. To know which application created the lead, check the lead source code.

 **Note:** The source code for LMA leads is `Package Installation`.

Let's tweak our example to see how multiple installation leads can be created for the same package. First, a user clicks **Get It Now**, and starts but doesn't complete the installation. AppExchange creates a lead with source code `SFDC-IN|Simple Sample App`. Later, the same user purchases your solution and installs it via an installation URL. The LMA creates a second lead with source code `Package Installation`. Same user. Same package. On the surface, the leads appear to be duplicates, but the lead source codes show that they aren't.

Learn more about LMA leads in [Lead and License Records in the LMA](#) on page 311.

## Duplicate Leads

A duplicate lead is a lead that AppExchange already sent to your org for this user, listing, or action within the past 180 days.

Duplicate lead source codes always contain the string `-dup-` and use the format `SFDC-dup-XX|Listing Name`. For example, `SFDC-dup-DM|Simple Sample App` indicates a duplicate lead from a user who clicked **View Demo** on the Simple Sample App listing.

## Troubleshoot AppExchange Leads

You enabled lead collection for your AppExchange listing. However, the lead count in your org is different than you expect. Learn how lead routing rules, reCaptcha verification, and other settings determine which leads AppExchange sends to your Salesforce org.

## Custom Lead Routing Rules

Typically, you set up custom lead routing rules to prevent duplicate or unwanted leads from reaching your sales team.

For example, an employee at your company watches your AppExchange listing's demo video. When prompted for contact information, they enter a company email address. AppExchange records this interaction as a lead. From a sales perspective, it's an unwanted lead.

You create a routing rule that prevents leads from users with your company's email address from propagating to your Salesforce org.

## Customer Contact Preferences

Customer can choose to share their contact info with, and allow contact from, AppExchange providers. AppExchange sends only leads to your Salesforce org for customers who allow provider contact.

## Web-to-Lead reCaptcha

To receive AppExchange leads in your Salesforce org, disable Require reCaptcha Verification in your org's Web-to-Lead settings. If reCaptcha is enabled, AppExchange leads aren't sent to your org.

**SETUP Web-to-Lead**

**Web-to-Lead Setup** [Help for this Page](#)

Using pre-existing pages on your company's website, you can capture contact and profile information from users and automatically generate new leads in salesforce.com, enabling you to respond in real-time to customer requests.

**My Website**

First Name

Last Name

E-mail

**SUBMIT**

**Create New Lead**

**Web-to-Lead Settings** [Edit](#) [Create Web-to-Lead Form](#)

Web-to-Lead Enabled ☒

**Require reCAPTCHA Verification** ☐

Default Lead Creator Sammy Sales

Default Response Template

## State and Country/Territory Picklists

AppExchange sends leads to your org via Web-to-Lead. Users provide contact info for the lead by completing the AppExchange Web-to-Lead form. They're required to select a country or territory from a picklist. The selected country or territory is saved as a text value. For example, a user selects Japan. The saved value is the full name of the country, Japan. The AppExchange lead is sent to your org with country set to Japan.

In orgs with state and country/territory picklists enabled, you optionally can populate these picklists with predefined, standard state and country lists that Salesforce provides. You can also edit country names and integration values, also known as developer names.

The Web-to-Lead form uses the integration values from the state and country/territory picklists. For AppExchange lead creation to succeed, the integration value for a country/territory in your org must match the value captured on the AppExchange Web-to-Lead form. In our example, they must both be Japan.

[Data Management](#) > [State and Country/Territory Picklists](#) > [Configure States and Countries and Territories](#)

## Country/Territory Details

---

### Country/Territory Information

Country/Territory Name

Country/Territory Code

Integration Value

Active ☒

Visible ☒

---

### States (0)

No records to display

Changing country/territory names doesn't affect AppExchange lead creation, but changing integration values does. Don't change integration values. The country or territory sent in an AppExchange lead must match an integration value in your org. If there's no match, lead creation fails. The same issue occurs with state picklists.

To avoid state and country/territory picklist-related lead failures, you have two options. Use the standard picklist integration values, or add duplicate states and countries/territories to your picklists.

### Use Standard Picklist Integration Values

To implement this option, use the Salesforce standard state and country/territory picklists in your org, and leave the integration values as-is. We recommend this option for most partners.

With this option, AppExchange leads propagate to your org with full state and country/territory names. The names match integration values in the standard picklists.

### Add Duplicate States and Countries/Territories to Your Picklists

Implement this option if you require two-letter state or country/territory abbreviations in your org. For example, you show abbreviations in the user interface, or use them to integrate with other systems.

Add duplicate states and countries/territories to your picklists with different integration values. Set one value to the two-letter state or country/territory abbreviation. Set the other value to the full state or country/territory name. Make only the two-letter abbreviation picklist entries visible.

With this option, AppExchange leads propagate to your org with full state and country/territory names, which match the full name integration values in your org. You also have two-letter integration values to use as needed.

### SEE ALSO:

[Standard Countries for Address Picklists](#)

[List of States and Countries Available from Data.com](#)

[Integration Values for State and Country Picklists](#)

# 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
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

Report	Metrics
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 Your AppExchange Listing

If you add features to a published solution, update your AppExchange listing so that new customers get access to the latest version. You can associate only an approved package version with your public listing. If your solution passed the security review within the last year, the new version is auto-approved. The package version must use the same namespace as the version that passed the review.



**Note:** We conduct periodic re-reviews of all solutions listed on AppExchange. If your solution shows significant change, we contact you to arrange a review of the new version.

1. Upload the new version of your package to the Publishing Console.
2. Log in to the [Salesforce Partner Community](#).
3. Click the **Publishing** tab.
4. Click the **Packages** tab. If you developed the new package in the same org as the previous version, the new package displays automatically. If you developed the new package in a different org, first navigate to the Organizations tab and connect the org that contains the package.
5. Find the new package, and then click **Start Review**.
6. If you're prompted to continue, click **Next**. You're guided through the options and settings that require your input. Update information as needed.
7. Click **Submit**. If your solution 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.
8. After your package is approved, navigate to the Listings tab and select the listing that you want to edit.
9. Click the solution-type tab. The tab name corresponds to the type of solution that you're listing: App, Component, Flow, and so on.
10. Click **Select Package**, and then select the new package you want to associate with the listing.
11. Save your changes.

## 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 Solution 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 Solution 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 find your solution or consulting service in several ways. On AppExchange, they search using keywords or browse using categories. They also find your listing via external search providers such as Google. Knowing how your listing is ranked in each of these scenarios helps you get the most visibility with potential customers.

### Keywords

Most of the time, customers look for solutions and consulting services by searching for a term, also known as a keyword, on AppExchange. 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 if it appears earlier in the listing.
- Generally, a keyword's relevance is increased if it appears more than once in the listing. Listing a keyword several times doesn't 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.

### Popularity

When customers look for solutions 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 actions customers take, such as watching a demo video and clicking the Learn More link. Activities that show greater commitment, such as installing a solution, are weighted more heavily than activities showing less commitment, such as clicking screenshots. The number of reviews and the average rating on a listing don't contribute to popularity.

### Sorting

Customers also sort results by rating, Salesforce edition, price, and other attributes. 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 one review and five stars is ranked above a listing with 20 four-star reviews.

### Search Engines

Because 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.

SEE ALSO:

[How Does AppExchange Search Work?](#)

## 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](http://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 328.

## 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 Sell on AppExchange with Checkout

### In this chapter ...

- AppExchange Checkout
- Checkout Management App

Bring a modern online shopping experience to your AppExchange listing with Checkout. Transform your Checkout data into insights and actions with the Checkout Management App (CMA).

## AppExchange Checkout

Checkout is AppExchange's integrated payments platform. With Checkout, customers can buy your AppExchange solution directly from your listing with a credit card or bank payment. Checkout is also ready to use with the License Management App (LMA) and the Checkout Management App (CMA).



**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>.

Here's how Checkout makes it easier to sell a solution on AppExchange.

You're interested in:	Checkout:
A modern and flexible payment experience.	Is built on Stripe, the industry leader in online payments.  You can accept credit cards, bank payments, or both. You can offer coupons and trials. You can also collect value-added tax (VAT) and US sales tax.
Automated licensing for your solution.	Is ready to use with the LMA.  When a customer purchases your solution using Checkout, a license record is automatically provisioned in the LMA. If a customer upgrades, renews, or cancels their subscription, Checkout updates the license.
Insights about your customers.	Is ready to use with the CMA.  The CMA brings the power of Salesforce CRM to Checkout. Use the CMA's dashboards to explore revenue, subscription status, and other key data. Send customizable notifications to customers and team members for trial expirations, declined payments, and other events.



**Tip:** Just getting started with Checkout? Head to Trailhead and earn the [AppExchange Checkout](#) badge.

### How Is Revenue Shared in AppExchange Checkout?

The revenue that you share with Salesforce depends on the payment type. If the customer pays with a bank transfer, the revenue share is 15%. If the customer pays with a credit card, the revenue share is 15%, plus a \$0.30 per transaction fee charged by our payment partner, Stripe. Regardless of the payment type, there's no minimum revenue share. We also don't charge setup fees, monthly service charges, or card storage fees.

### Payment Plans in AppExchange Checkout

Checkout supports two types of payment plans: one-time and subscription. For either type of plan, you can charge customers on a per user or per company basis. If you charge on a per user basis, your customer buys an individual license for every user in their org who uses your solution. If you charge on a per company basis, your customer buys an org-wide license. An org-wide license means that every user in their org can use your solution. To provide customers with flexible payment options, you can combine multiple plans on your listing.

### Payment Methods in AppExchange Checkout

Checkout supports two payment methods: credit cards and bank account transfers. You can accept one or both payment methods on your listing.

[Get Started with AppExchange Checkout](#)

To begin accepting payments with Checkout, first create a Stripe account. If you plan to offer a subscription for your solution, configure a product and pricing plan in Stripe. Then, enable Checkout on your AppExchange listing and choose payment plans and methods.

[Support International Payments in AppExchange Checkout](#)

In a few steps, you can get Checkout ready to accept payments from customers in the European Union (EU) and other regions. First, verify that your company is based in a country that's supported by our payment partner, Stripe. Then, if your country's tax authority requires you to collect Value Added Tax (VAT), enable VAT in the Publishing Console.

[Manage AppExchange Checkout Subscriptions](#)

Handle common customer requests related to Checkout subscriptions, such as viewing payment history, adding or removing licenses, and canceling subscriptions.

[AppExchange Checkout FAQs](#)

Find answers to common questions about Checkout.

[AppExchange Checkout Considerations](#)

Keep these considerations in mind when using Checkout.

## How Is Revenue Shared in AppExchange Checkout?

The revenue that you share with Salesforce depends on the payment type. If the customer pays with a bank transfer, the revenue share is 15%. If the customer pays with a credit card, the revenue share is 15%, plus a \$0.30 per transaction fee charged by our payment partner, Stripe. Regardless of the payment type, there's no minimum revenue share. We also don't charge setup fees, monthly service charges, or card storage fees.

To see how revenue sharing works, let's look at some examples.

Payment Type	Example
Bank transfer	<p>You sell an app for \$50 per user per month. If a customer buys 10 licenses with a bank transfer, here's how revenue is shared.</p> <ul style="list-style-type: none"><li>• The overall transaction amount is \$500 per month (\$50 per user per month x 10 users).</li><li>• The amount shared with Salesforce is \$75 per month (15% x \$500 per month).</li></ul>
Credit card	<p>You sell an app for \$1,000 per user per year. If a customer buys 5 licenses with a credit card, here's how revenue is shared.</p> <ul style="list-style-type: none"><li>• The overall transaction amount is \$5,000 per year (\$1,000 per user per year x 5 users).</li><li>• The amount shared with Salesforce is \$750.00 per year (15% x \$5,000 per year).</li><li>• The amount shared with Stripe is \$0.30 (1 credit card transaction x \$0.30 per transaction fee).</li></ul>

## Payment Plans in AppExchange Checkout

Checkout supports two types of payment plans: one-time and subscription. For either type of plan, you can charge customers on a per user or per company basis. If you charge on a per user basis, your customer buys an individual license for every user in their org who uses your solution. If you charge on a per company basis, your customer buys an org-wide license. An org-wide license means that every user in their org can use your solution. To provide customers with flexible payment options, you can combine multiple plans on your listing.

Here's a breakdown of the payment plans that you can offer.

Plan	Pricing Options	Customer is billed:	Set up the plan in:
One-time	<ul style="list-style-type: none"> <li>Per User</li> <li>Per Company</li> </ul>	Once, at the time of purchase	The Publishing Console on the Salesforce Partner Community
Subscription	<ul style="list-style-type: none"> <li>Per User</li> <li>Per Company</li> </ul>	On a recurring basis, either monthly or annually	The Stripe dashboard

To provide customers with the most flexibility, we recommend offering several payment options on a listing.

## Payment Methods in AppExchange Checkout

Checkout supports two payment methods: credit cards and bank account transfers. You can accept one or both payment methods on your listing.

Payment Method	Customers pay with:	Notes
Credit card	Visa, MasterCard, American Express, JCB, Discover, or Diners Club credit cards.	Payments are processed immediately.
US bank account	Checking, savings, or money market accounts from banks based in the United States. Payments are processed using the Automated Clearing House (ACH) network.	<ul style="list-style-type: none"> <li>Payments can take up to 5 days to process.</li> <li>Your pricing plan in Stripe must be in US dollars (USD).</li> <li>Customers must pay with a business bank account. Checkout doesn't support ACH payments from personal bank accounts.</li> <li>Customers must have a US billing address.</li> </ul>
European bank account	Checking, savings, or money market accounts from banks based in the European Union. Payments are processed using the Single Euro Payment Area (SEPA) framework.	<ul style="list-style-type: none"> <li>Payments are processed immediately.</li> <li>Your pricing plan in Stripe must be in euros (EUR).</li> <li>Customers must have an EU billing address.</li> </ul>



**Note:** Your business address 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.

## Get Started with AppExchange Checkout

To begin accepting payments with Checkout, first create a Stripe account. If you plan to offer a subscription for your solution, configure a product and pricing plan in Stripe. Then, enable Checkout on your AppExchange listing and choose payment plans and methods.

### Create a Stripe Account for AppExchange Checkout

Before you enable Checkout on a listing, create an account with our payment partner, Stripe.

### Create a Stripe Product and Pricing Plan for AppExchange Checkout

To offer a subscription of your solution in Checkout, first create a product and pricing plan in your Stripe dashboard. A product represents the solution or service that you sell. A pricing plan sets the product's cost, currency, and billing frequency.

### Activate Bank Payments for AppExchange Checkout

To let customers pay for your solution with a bank transfer, 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.

### Enable Checkout on an AppExchange Listing

After you create a Stripe account and set up pricing plans for your solution, you can enable Checkout on an AppExchange listing. When you enable Checkout, you choose the payment plans and methods that you support.

### Send Email Receipts for AppExchange Checkout Purchases

To send customers receipts for Checkout purchases, set up email receipts in your Stripe dashboard.

### Preview the AppExchange Checkout Experience

If you've enabled Checkout on your listing, you can preview the customer purchase experience by modifying the AppExchange listing URL.

### Convert a Free Listing to Use AppExchange Checkout

If the solution associated with your free AppExchange listing passed security review, you can convert the listing to accept payments using 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 solution.

## Create a Stripe Account for AppExchange Checkout

Before you enable Checkout on a listing, create an account with our payment partner, Stripe.

Before you create your account, have the following information available.

- A short description of your business, such as the products you sell
- Basic information about your business, like its physical address
- 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 payments

After you've gathered this information, you're ready to go.

1. Log in to the [Salesforce 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, sign in instead.

After you create the account, you can manage it on the Stripe website. To learn more about Stripe, go to <https://stripe.com/docs/dashboard>.

### USER PERMISSIONS

To manage AppExchange listings:

- **Manage Listings**

## Create a Stripe Product and Pricing Plan for AppExchange Checkout

To offer a subscription of your solution in Checkout, first create a product and pricing plan in your Stripe dashboard. A product represents the solution or service that you sell. A pricing plan sets the product's cost, currency, and billing frequency.

You can create multiple pricing plans for your product. For example, you can create one plan that uses monthly billing and another plan that uses annual billing.

1. Log in to [Stripe](#).
2. From your Stripe dashboard, click **Billing > Products**.
3. Click **New**.
4. Provide a name for your product, and then click **Create product**.
5. If prompted, enter your Stripe password, and then click **Authenticate**.
6. Provide a nickname for your plan.



**Tip:** Include the billing frequency, such as *Annual*, in your plan's nickname.

7. Select a currency.



**Note:** To let customers pay with US bank accounts, use US dollars (USD) for the plan. To let customers pay with European bank accounts, use euros (EUR) for the plan.

8. Set a unit price.

In the Publishing Console, you specify whether to apply this unit price per user or per company (org-wide).

9. Specify a monthly or yearly billing interval.
10. Click **Add pricing plan**.

If you've connected your Stripe account to the Publishing Console, your pricing plans are ready to add to your listing.

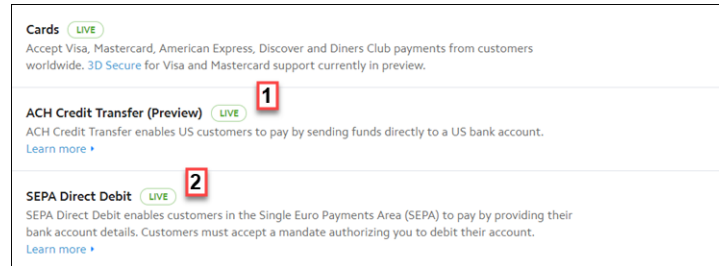
## Activate Bank Payments for AppExchange Checkout

To let customers pay for your solution with a bank transfer, 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 business address 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. Click **Settings**.
4. Under Payments and Payouts, click **Payment methods**.
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 Checkout on an AppExchange Listing

After you create a Stripe account and set up pricing plans for your solution, you can enable Checkout on an AppExchange listing. When you enable Checkout, you choose the payment plans and methods that you support.

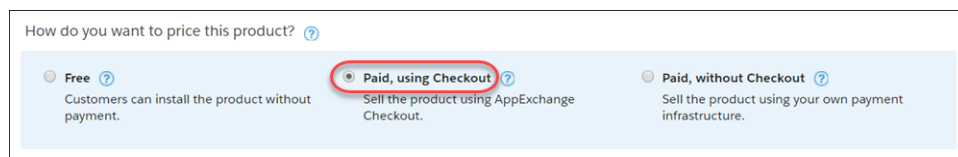
Before you enable Checkout on a listing, verify that Salesforce approved your business plan.

1. Log in to the [Salesforce Partner Community](#).
2. Click **Publishing**.
3. Create a listing, or edit an existing one.
4. On the Pricing tab, select **Paid, using Checkout**.

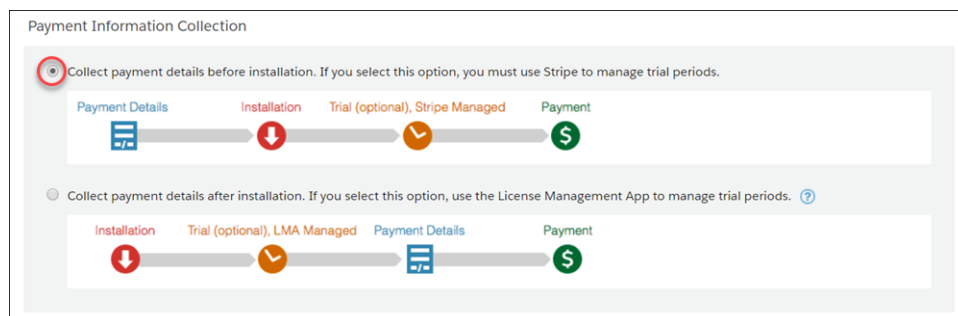
### USER PERMISSIONS

To manage AppExchange listings:

- [Manage Listings](#)




5. Select when to collect payment details from the customer.



6. Select payment plans (1) and adjust the display order (2).

Subscription Plans [?](#) 2 [Manage Plans in Stripe](#)

List Plan?	Name	Trial	Price	Units	Display Order
<input type="checkbox"/>		0 days	€3,304 EUR		
<input type="checkbox"/>		30 days	\$10 USD		
<input checked="" type="checkbox"/>		0 days	\$199.99 USD	User	2
	\$199.99 USD per user per month				
<input checked="" type="checkbox"/>		0 days	€123 EUR	User	1 (Default)
	€123 EUR per user per month				


 **Tip:** You can offer multiple payment plans on a listing. For example, you can offer a one-time payment option and a monthly subscription.


Payment Plan	Steps
Subscription	<div>a. Select one of the pricing plans that you created in Stripe.</div> <div>b. Select company or per-user pricing.</div>
One-time payment	<div>a. Click <b>Add One Time Price Option</b>.</div> <div>b. Provide a name for the plan.</div> <div>c. Select a price and currency.</div> <div>d. Select company or per-user pricing.</div>


7. Select the payment methods that you accept.

Payment Methods [?](#)


Customers can pay for this listing with:

☒ **Credit Cards**  
 Includes Visa, MasterCard, American Express, JCB, Discover, or Diners Club.

☐ **US Bank Accounts**  
 Includes checking, savings, and money market accounts.

☒ **European Bank Accounts**  
 Includes checking, savings, and money market accounts.

Creditor Identifier

 **Note:** Your billing address 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**.

SEE ALSO:

- [Create a Stripe Product and Pricing Plan for AppExchange Checkout](#)
- [Enable Checkout on an AppExchange Listing](#)

## Send Email Receipts for AppExchange Checkout Purchases

To send customers receipts for Checkout purchases, set up email receipts in your Stripe dashboard.

1. Log in to [Stripe](#).
2. From your Stripe dashboard, click **Settings**.
3. Under Payments and Payouts, click **Email receipts**.
4. Enable the setting for successful payments.
5. Click **Save**.

## Preview the AppExchange Checkout Experience

If you've enabled Checkout on your listing, you can preview the customer purchase experience by modifying the AppExchange listing URL.

1. Go to your solution's AppExchange listing.
2. Append `&modal=appx_getitnow_buyform_modal` to the listing URL, and then refresh the page.

## Convert a Free Listing to Use AppExchange Checkout

If the solution associated with your free AppExchange listing passed security review, you can convert the listing to accept payments using 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 solution.

1. Log in to the [Salesforce 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. In the case description, explain that you're converting a free listing to paid using AppExchange Checkout. Include your listing's URL.

### USER PERMISSIONS

To manage AppExchange listings:

- [Manage Listings](#)

- e. Click **Submit Case**.

Salesforce will contact you to arrange for payment of the security review fee.

## Support International Payments in AppExchange Checkout

In a few steps, you can get Checkout ready to accept payments from customers in the European Union (EU) and other regions. First, verify that your company is based in a country that's supported by our payment partner, Stripe. Then, if your country's tax authority requires you to collect Value Added Tax (VAT), enable VAT in the Publishing Console.

### [Verify AppExchange Checkout Is Supported in Your Country](#)

To accept payments with Checkout, your company must be based in a country that's supported by our payment partner, Stripe.

### [Collect VAT for AppExchange Checkout Transactions](#)

If your country's tax authority requires you to collect Value-Added Tax (VAT), you can include VAT in 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.

### [Strong Customer Authentication for AppExchange Checkout](#)

Strong customer authentication (SCA) enhances the security of online payments with an identity verification step. Learn how SCA works, which regions require it, and how it affects Checkout payments. Then get your company and customers ready for SCA.

## Verify AppExchange Checkout Is Supported in Your Country

To accept payments with Checkout, your company must be based in a country that's supported by our payment partner, Stripe.

1. To see a list of supported countries, go to <https://stripe.com/global>.  
If your country is listed, you're eligible to use Checkout.
2. If Stripe isn't supported in the country where your company is based, sign up to get notified when it's available.

## Collect VAT for AppExchange Checkout Transactions

If your country's tax authority requires you to collect Value-Added Tax (VAT), you can include VAT in 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.

1. Log in to the [Salesforce 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. Use the same country that you provided to Stripe in your billing address.
6. Click **Save**.

If you manage Checkout data with the Checkout Management App, you can use the app to view information for VAT reporting.

### USER PERMISSIONS

To manage AppExchange listings:

- [Manage Listings](#)

## Strong Customer Authentication for AppExchange Checkout

Strong customer authentication (SCA) enhances the security of online payments with an identity verification step. Learn how SCA works, which regions require it, and how it affects Checkout payments. Then get your company and customers ready for SCA.

### What Is Strong Customer Authentication?

Strong customer authentication (SCA) enhances the security of online payments with an identity verification step. SCA is required for online payments in the European Economic Area, including AppExchange Checkout payments.

### How Strong Customer Authentication Affects AppExchange Checkout

Strong customer authentication (SCA) is automatically integrated into the Checkout payment experience for European customers. Learn how SCA affects the initial purchase and recurring payments.

### Strong Customer Authentication Best Practices for AppExchange Checkout

If you sell an AppExchange solution in a region that requires strong customer authentication (SCA), follow these Checkout best practices.

## What Is Strong Customer Authentication?

Strong customer authentication (SCA) enhances the security of online payments with an identity verification step. SCA is required for online payments in the European Economic Area, including AppExchange Checkout payments.

SCA is mandated by the Second Payment Services Directive (PSD2), which introduces laws to enhance the security of online payments in the European Economic Area. Starting on September 14, 2019, customers who live in this region may be asked to perform an identity verification step to make purchases online.

A customer can verify their identity with a password, a code delivered to a mobile device, or using biometric data, such as a fingerprint. This verification step applies to one-time purchases and recurring payments, such as subscriptions. The customer's bank or credit card issuer determines when to request that the customer authorize the purchase by verifying their identity.

Starting on September 14, 2019, Checkout automatically integrates SCA into the payment experience for European customers. To learn more about SCA, go to <https://stripe.com/docs/strong-customer-authentication>.

## How Strong Customer Authentication Affects AppExchange Checkout

Strong customer authentication (SCA) is automatically integrated into the Checkout payment experience for European customers. Learn how SCA affects the initial purchase and recurring payments.

### Initial Purchase

The initial purchase is your customer's first Checkout transaction. In the initial purchase, the customer uses the Checkout wizard to select a payment plan and method, provide billing and contact information, and confirm the payment. In regions that require SCA, the Checkout wizard adds an identity verification step.

After the customer clicks **Purchase**, Checkout may prompt them to verify their identity. For example, they might be asked to enter a verification code that's sent to the mobile device associated with their payment method. This verification step uses the 3D Secure 2 protocol and is managed by Checkout's payment partner, Stripe. After the customer verifies their identity, Stripe processes the payment.

AppExchange Checkout

Confirm your order

One last thing: Double check your order details, and then agree to our terms and conditions. If you're purchasing a subscription, you can edit it anytime on the My Installs and Subscriptions page of your AppExchange profile.

Payment Plan  
\$10 USD per user per month

\*Users  
10

Coupon Code  
FABISOFOREVER

Subtotal  
\$100.00  
Discount  
-50%  
US Sales Tax  
\$4.63\*

Payment (Monthly)  
\$54.63\*

\* Subject to change based on shipping address and tax changes.

Start Date  
2/12/2021  
Provider  
Northern Star

Payment Method  
Credit Card

Contact Information  
Dee Mato  
dmato@example.com

Billing Address  
1 Main Birch Ter  
Fremont, CA 94536, US

Shipping Address  
1 Main Birch Ter  
Fremont, CA 94536, US

Your bank or credit card issuer may require you to authorize future payments. If your authorization is required, the provider will notify you.

Back

Purchase

Recurring Payments

Customers can also make recurring payments, either monthly or annually. In regions that require SCA, the first payment is the initial purchase, and the customer may be asked to verify their identity to complete the transaction. Checkout attempts to process subsequent payments with the billing details provided in the initial purchase, per the terms and conditions of the subscription. In regions that require SCA, the customer’s bank or credit card issuer reviews the payment attempt and determines whether to request customer authorization. If customer authorization is required, Stripe marks the payment as failed. The next time the customer logs in to AppExchange, Checkout prompts the customer to authorize the payment (1). The customer clicks **Authorize** (2), then verifies their identity using the same process as the initial purchase. After the customer verifies their identity, Stripe processes the payment.

1 Authorize your AppExchange subscription payment

We tried to process a payment for an AppExchange subscription, but your bank or credit card issuer requires your authorization.

To continue the subscription, review the following details and authorize the payment of \$36.00. When you authorize the payment, you'll be prompted by your bank or credit card issuer to verify your identity.

Subscription Details

Solution  
Appy's Maps

Provider  
Appy's Geospatial Solutions, LLC

Payment Plan  
\$12 USD per user per month

Users  
3

Monthly Payment  
\$36.00

Payment Method  
Visa ending in

Cancel

2 Authorize

Strong Customer Authentication Best Practices for AppExchange Checkout

If you sell an AppExchange solution in a region that requires strong customer authentication (SCA), follow these Checkout best practices.

1. Prepare Your Customers for Strong Customer Authentication

If you serve customers in the European Economic Area, communicate how strong customer authentication (SCA) affects online payments, including payments for your AppExchange solution.

190

2. [Manage AppExchange Checkout Subscription Payments That Require Customer Authorization](#)

If a Checkout subscription payment fails because it requires customer authorization, determine how Stripe handles the related subscription. For example, you can configure Stripe to cancel the subscription, mark the subscription as unpaid, or take no action.

3. [View AppExchange Checkout Subscription Payments That Require Customer Authorization](#)

If a Checkout subscription payment can't be processed because it requires customer authorization, Stripe marks the payment as failed. View these payments in the Stripe dashboard to see transaction details, including customer contact information. You can use this information to follow up with the customer and provide instructions for authorizing the payment on AppExchange.

4. [Authorize an AppExchange Checkout Subscription Payment](#)

In regions that require strong customer authentication (SCA), a customer's bank or credit card issuer may require the customer to authorize Checkout subscription payments periodically. To see payments that require customer authorization, check your Stripe dashboard. If authorization is required, we prompt the customer when they log in to AppExchange. However, you can also provide customers with self-service instructions for authorizing a payment.

## Prepare Your Customers for Strong Customer Authentication

If you serve customers in the European Economic Area, communicate how strong customer authentication (SCA) affects online payments, including payments for your AppExchange solution.

In your communication, we recommend that you:

- Define SCA and explain how SCA changes the online payment experience.
- Note that SCA impacts many types of online payments in the European Economic Area, including AppExchange payments.
- Explain that the customer may be asked to authorize AppExchange payments periodically, which includes an identity verification step.
- Explain that if authorization is required, we prompt the customer when they log in to AppExchange.
- Provide self-service steps for authorizing an AppExchange subscription payment.

### SEE ALSO:

[Manage AppExchange Checkout Subscription Payments That Require Customer Authorization](#)

[View AppExchange Checkout Subscription Payments That Require Customer Authorization](#)

[Authorize an AppExchange Checkout Subscription Payment](#)

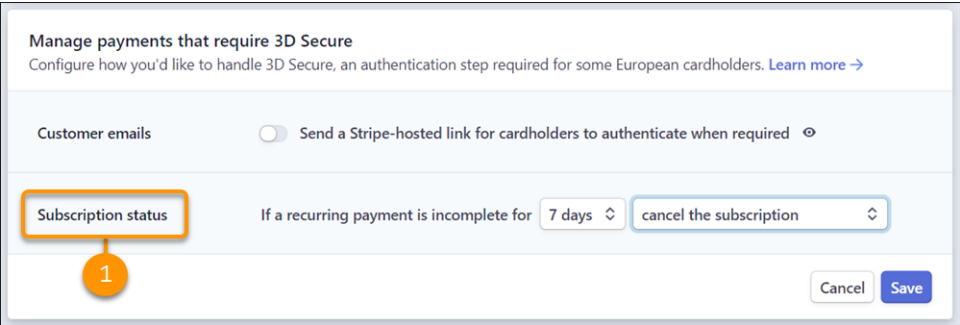
## Manage AppExchange Checkout Subscription Payments That Require Customer Authorization

If a Checkout subscription payment fails because it requires customer authorization, determine how Stripe handles the related subscription. For example, you can configure Stripe to cancel the subscription, mark the subscription as unpaid, or take no action.

1. Log in to [Stripe](#).
2. From your Stripe dashboard, click **Settings**.
3. Under Billing, click **Subscriptions and emails**.
4. Go to Manage payments that require 3D Secure, and then configure `Subscription status (1)`.



**Important:** Don't enable the `Customer emails` setting. To authorize payments, customers must log in to AppExchange.



View AppExchange Checkout Subscription Payments That Require Customer Authorization

If a Checkout subscription payment can't be processed because it requires customer authorization, Stripe marks the payment as failed. View these payments in the Stripe dashboard to see transaction details, including customer contact information. You can use this information to follow up with the customer and provide instructions for authorizing the payment on AppExchange.

- 1. Log in to [Stripe](#).
- 2. From your Stripe dashboard, click **Payments**.
- 3. Configure the payment filters as follows.

Filter	Value
Status	Incomplete

- 4. Click **Done**.
- 5. Click a payment to view details about the transaction.

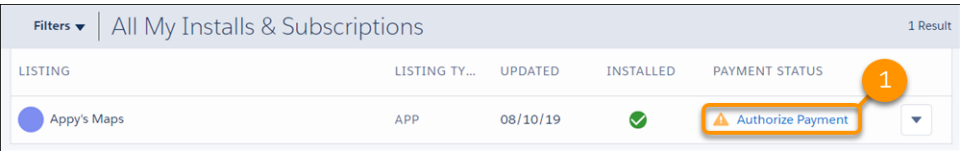
Authorize an AppExchange Checkout Subscription Payment

In regions that require strong customer authentication (SCA), a customer's bank or credit card issuer may require the customer to authorize Checkout subscription payments periodically. To see payments that require customer authorization, check your Stripe dashboard. If authorization is required, we prompt the customer when they log in to AppExchange. However, you can also provide customers with self-service instructions for authorizing a payment.

USER PERMISSIONS

- To manage AppExchange subscriptions:
- [Manage Billing](#)

- 1. Log in to [AppExchange](#).
- 2. From the user profile menu, click **My Installs & Subscriptions**.
- 3. Find the subscription that requires authorization.
- 4. Click **Authorize Payment** (1).



5. Review the subscription details, and then click **Authorize**.

SEE ALSO:

[View AppExchange Checkout Subscription Payments That Require Customer Authorization](#)

## Manage AppExchange Checkout Subscriptions

Handle common customer requests related to Checkout subscriptions, such as viewing payment history, adding or removing licenses, and canceling subscriptions.

### [View AppExchange Checkout Receipts](#)

If a customer requests a receipt for a previous Checkout payment, you can share self-service steps for viewing payment history on AppExchange.

### [Add or Remove Licenses from an AppExchange Checkout Subscription](#)

Your customers can add or remove licenses from their Checkout subscriptions on AppExchange. If a customer adds licenses during the current billing period, the additional licenses are available immediately. Checkout charges the customer a prorated amount for their next billing period. If a customer removes licenses, the removal takes effect at the start of their next billing period. Checkout charges the customer for the reduced license count when the removal takes effect. Share these self-service steps for updating their subscription on AppExchange.

### [Cancel an AppExchange Checkout Subscription](#)

If a customer wants to end a Checkout subscription, you can share self-service steps for canceling the subscription on AppExchange. The cancellation takes effect at the end of the contract term.

## View AppExchange Checkout Receipts

If a customer requests a receipt for a previous Checkout payment, you can share self-service steps for viewing payment history on AppExchange.

1. Log in to [AppExchange](#).
2. From the user profile menu, click **My Installs & Subscriptions**.
3. Find the subscription whose payment history you want to view.
4. From the dropdown list, select **Manage Subscription**.
5. Go to Payment History, and then click an invoice to view details about the purchase.

### USER PERMISSIONS

To manage AppExchange subscriptions:

- **Manage Billing**

## Add or Remove Licenses from an AppExchange Checkout Subscription

Your customers can add or remove licenses from their Checkout subscriptions on AppExchange. If a customer adds licenses during the current billing period, the additional licenses are available immediately. Checkout charges the customer a prorated amount for their next billing period. If a customer removes licenses, the removal takes effect at the start of their next billing period. Checkout charges the customer for the reduced license count when the removal takes effect. Share these self-service steps for updating their subscription on AppExchange.

### USER PERMISSIONS

To manage AppExchange subscriptions:

- **Manage Billing**



**Warning:** Don't use the Stripe website to change the number of seats included in your AppExchange customers' licenses. The changes won't sync to Checkout or the License

Management App (LMA). Instead, make license updates with the LMA using the instructions in [Modify a License Record](#).

1. Log in to [AppExchange](#).
2. From the user profile menu, click **My Installs & Subscriptions**.
3. Find the subscription that you want to update.
4. From the dropdown list, select **Manage Subscription**.
5. Click **Edit**.
6. Go to Payment Details, and then edit the number of licenses associated with the subscription.
7. Click **Review Changes**.
8. Agree to the terms and conditions, and then click **Save**.

## Cancel an AppExchange Checkout Subscription

If a customer wants to end a Checkout subscription, you can share self-service steps for canceling the subscription on AppExchange. The cancellation takes effect at the end of the contract term.

1. Log in to [AppExchange](#).
2. From the user profile menu, click **My Installs & Subscriptions**.
3. Find the subscription that you want to cancel.
4. From the dropdown list, select **Manage Subscription**.
5. Click **End Subscription**, and then confirm the cancellation.

### USER PERMISSIONS

To manage AppExchange subscriptions:

- [Manage Billing](#)

## AppExchange Checkout FAQs

Find answers to common questions about Checkout.

### [Does AppExchange Checkout replace the License Management App?](#)

No, Checkout works with the LMA to support the licensing process. When a customer purchases your solution, Checkout creates a license record in the LMA. If a customer edits their subscription on AppExchange, such as by adding seats, the license record in the LMA automatically updates to reflect those changes.

### [How does AppExchange Checkout affect Trialforce and lead management?](#)

Checkout doesn't affect your Trialforce configuration or how you manage leads. However, when a customer signs up for a trial using Checkout, the corresponding trial user is listed as `Active` in the License Management App (LMA).

### [Should I collect payment details from AppExchange Checkout customers before or after installation?](#)

Both approaches have advantages. We recommend thinking about your target customers and existing business processes, and then deciding.

### [Does AppExchange Checkout support multiple currencies?](#)

Yes. To offer another currency on your listing, first create a pricing plan in Stripe that uses this currency. Then, go to the Publishing Console and add the plan to your listing. When a customer purchases your solution, Checkout charges them in the currency that you specified on the plan. When Stripe transfers the payment to you, it's converted to the currency used by your bank account.

### [If I use AppExchange Checkout to sell my solution, do customers have to purchase from AppExchange?](#)

Yes, purchases must occur on AppExchange and are subject to revenue sharing per your Salesforce partnership agreement. Also, if the transaction is processed another way, Checkout can't associate the purchase with your solution or provision licenses with the License Management App (LMA).

### Can my customer switch to another AppExchange Checkout payment plan?

Yes, you can switch the customer to another plan in Stripe. The new plan takes effect at the start of the next billing period. If you want the change to take effect immediately, cancel the current plan in Stripe and ask the customer to purchase the new plan from your listing.

### If a customer's credit card payment is declined in AppExchange Checkout, does their license become inactive?

In your Stripe settings, you determine 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 AppExchange Checkout customers add or remove licenses during the current billing period?

If a customer adds licenses during the current billing period, the licenses are available for immediate use. Checkout charges the customer a prorated amount for their next billing period. If a customer removes licenses, the reduction takes effect at the start of their next billing period. The customer can continue to use the licenses during their current billing period. Checkout charges the customer for the reduced license count starting with their next billing period.

### If an admin purchases and installs a solution with AppExchange Checkout, can another user edit the subscription on AppExchange?

Yes, provided the user has the "Manage Billing" permission in the Salesforce org associated with the subscription.

### Does AppExchange Checkout support price tiers in Stripe?

No. If you add multiple price tiers to a pricing plan in Stripe, Checkout can't import the plan to the Publishing Console. If you're interested in providing discounted pricing for your solution, create a coupon in Stripe and share it with your customers.

### Why can't my customer make an AppExchange Checkout purchase?

If a customer clicks **Get It Now** on your listing, but can't make a Checkout purchase, verify that the customer is logged in to AppExchange with a supported Salesforce org. Checkout supports only paid orgs whose status is `Active`. Trial orgs, sandbox orgs, and Developer Edition orgs aren't supported.

### Does AppExchange Checkout support tax rates created in Stripe?

No. Although Stripe allows you to create tax rates, Checkout doesn't support the Stripe rates. Salesforce internally manages tax rates, including those for US sales tax and value-added tax (VAT).

## Does AppExchange Checkout replace the License Management App?

No, Checkout works with the LMA to support the licensing process. When a customer purchases your solution, Checkout creates a license record in the LMA. If a customer edits their subscription on AppExchange, such as by adding seats, the license record in the LMA automatically updates to reflect those changes.

## How does AppExchange Checkout affect Trialforce and lead management?

Checkout doesn't affect your Trialforce configuration or how you manage leads. However, when a customer signs up for a trial using Checkout, the corresponding trial user is listed as `Active` in the License Management App (LMA).

## Should I collect payment details from AppExchange Checkout customers before or after installation?

Both approaches have advantages. We recommend thinking about your target customers and existing business processes, and then deciding.

Here's a table that you can use to guide your decision.

<b>Collect payment details:</b>	<b>Best if your customers value:</b>	<b>Best if your company:</b>
<i>Before installation</i>	A seamless purchase experience at the end of a trial.	Prefers to manage trials in Stripe.
<i>After installation</i>	The ability to quickly try out your solution.	Prefers to manage trials using the License Management App (LMA).

## Does AppExchange Checkout support multiple currencies?

Yes. To offer another currency on your listing, first create a pricing plan in Stripe that uses this currency. Then, go to the Publishing Console and add the plan to your listing. When a customer purchases your solution, Checkout charges them in the currency that you specified on the plan. When Stripe transfers the payment to you, it's converted to the currency used by your bank account.

SEE ALSO:

[Create a Stripe Product and Pricing Plan for AppExchange Checkout](#)

## If I use AppExchange Checkout to sell my solution, do customers have to purchase from AppExchange?

Yes, purchases must occur on AppExchange and are subject to revenue sharing per your Salesforce partnership agreement. Also, if the transaction is processed another way, Checkout can't associate the purchase with your solution or provision licenses with the License Management App (LMA).

## Can my customer switch to another AppExchange Checkout payment plan?

Yes, you can switch the customer to another plan in Stripe. The new plan takes effect at the start of the next billing period. If you want the change to take effect immediately, cancel the current plan in Stripe and ask the customer to purchase the new plan from your listing.

## If a customer's credit card payment is declined in AppExchange Checkout, does their license become inactive?

In your Stripe settings, you determine 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 AppExchange Checkout customers add or remove licenses during the current billing period?

If a customer adds licenses during the current billing period, the licenses are available for immediate use. Checkout charges the customer a prorated amount for their next billing period. If a customer removes licenses, the reduction takes effect at the start of their next billing period. The customer can continue to use the licenses during their current billing period. Checkout charges the customer for the reduced license count starting with their next billing period.

## If an admin purchases and installs a solution with AppExchange Checkout, can another user edit the subscription on AppExchange?

Yes, provided the user has the “Manage Billing” permission in the Salesforce org associated with the subscription.

## Does AppExchange Checkout support price tiers in Stripe?

No. If you add multiple price tiers to a pricing plan in Stripe, Checkout can't import the plan to the Publishing Console. If you're interested in providing discounted pricing for your solution, create a coupon in Stripe and share it with your customers.

## Why can't my customer make an AppExchange Checkout purchase?

If a customer clicks **Get It Now** on your listing, but can't make a Checkout purchase, verify that the customer is logged in to AppExchange with a supported Salesforce org. Checkout supports only paid orgs whose status is `Active`. Trial orgs, sandbox orgs, and Developer Edition orgs aren't supported.

## Does AppExchange Checkout support tax rates created in Stripe?

No. Although Stripe allows you to create tax rates, Checkout doesn't support the Stripe rates. Salesforce internally manages tax rates, including those for US sales tax and value-added tax (VAT).


## AppExchange Checkout Considerations

Keep these considerations in mind when using Checkout.

- You must distribute your product as a managed package.
- You can't use Checkout with OEM apps.

## 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.

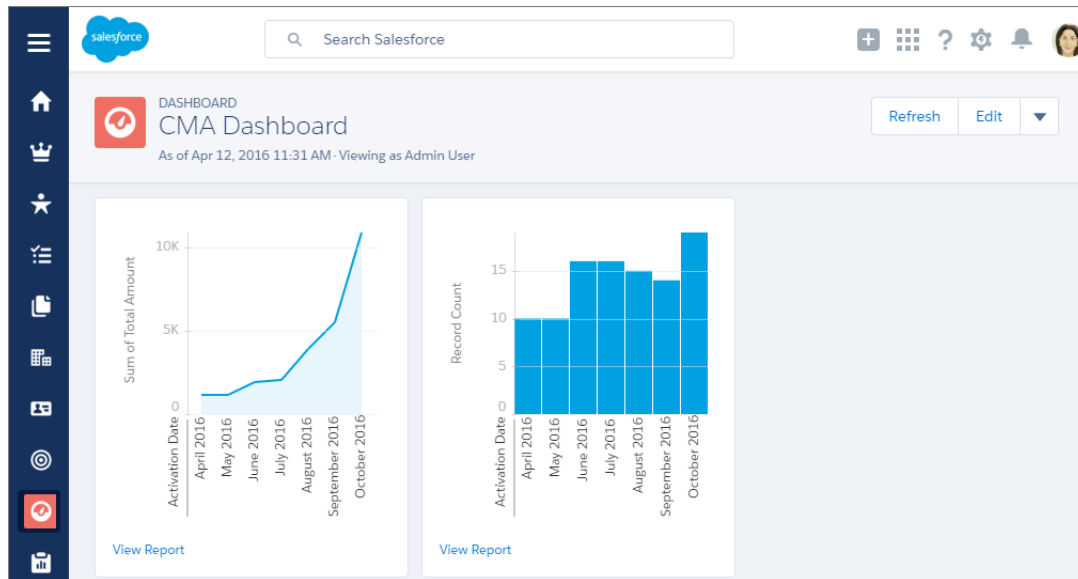
 **Note:** The CMA is available in English and Japanese to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, visit <https://partners.salesforce.com>.

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

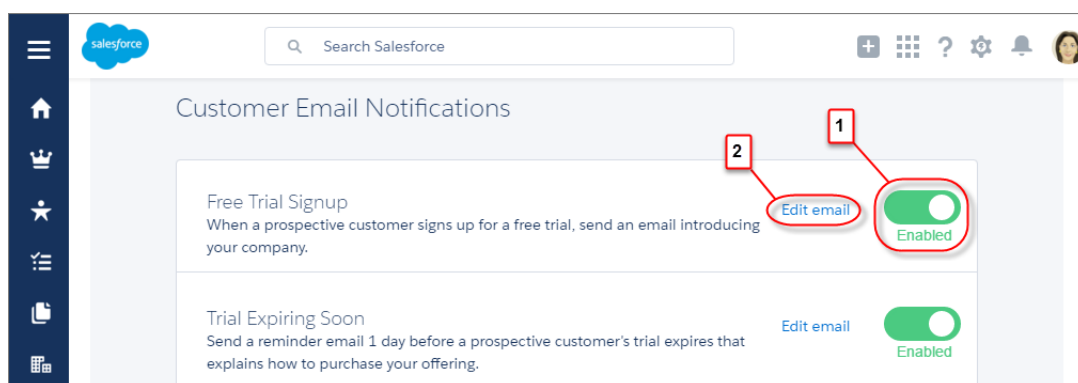


The dashboard is preconfigured to show:

- 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.

SEE ALSO:

[Modify a Notification Template in the Checkout Management App](#)

## 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: <ul style="list-style-type: none"><li>Subscription</li></ul>
Subscription (2)	Contains information about the customer’s history and usage of an offering. For example, when the subscription started.	Child object of: <ul style="list-style-type: none"><li>Subscription plan</li></ul> Parent object of: <ul style="list-style-type: none"><li>Invoice</li></ul>

Object	Purpose	Relationships
		<ul style="list-style-type: none"> <li>Transaction</li> </ul>
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: <ul style="list-style-type: none"> <li>Subscription</li> </ul> Sibling object of: <ul style="list-style-type: none"> <li>Transaction</li> </ul>
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: <ul style="list-style-type: none"> <li>Invoice</li> </ul>
Transaction (5)	Contains information about a customer payment attempt. For example, method of payment and whether it was successful.	Child object of: <ul style="list-style-type: none"> <li>Subscription</li> </ul> Sibling object of: <ul style="list-style-type: none"> <li>Invoice</li> </ul>

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.

### [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.

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience


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

### 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.

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=a0N3A000000rMcIUAE>.
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.

SEE ALSO:

[Assign Access to the Checkout Management App](#)

## 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.

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.

### USER PERMISSIONS

To install packages:


- Download AppExchange Packages

### USER PERMISSIONS

To configure the Checkout Management App:

- CMA Admin User

- 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.

#### SEE ALSO:

[Sample Checkout Management App Customizations](#)

## 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.

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**.

### USER PERMISSIONS


To assign a permissions set:

- Assign Permission Sets

## 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.

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**.

### USER PERMISSIONS

To enable, disable, or customize notifications:

- CMA Admin User

To create or change Visualforce email templates:

- Customize Application

SEE ALSO:

[Use an Organization-Wide Address on a Notification](#)

[Include a Link in a Notification](#)

## 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.

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 <div> <b>Note:</b> Namespace prefix: <code>sfcma</code></div> |
| 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**.
- ### USER PERMISSIONS
- To manage, create, edit, and delete custom settings:

  - Customize Application

To save changes to Apex classes and triggers:

  - Author Apex
- 205

## 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.

### 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.

### 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.

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.

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>*.

### USER PERMISSIONS

To enable, disable, or customize notifications:

- CMA Admin User

To configure organization-wide addresses:

- Modify All Data

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.

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**.

### USER PERMISSIONS

To enable, disable, or customize notifications:

- CMA Admin User

To create or change Visualforce email templates:

- Customize Application

## 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.

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	<i>Revenue Per Year</i> To keep your dashboard organized, include the name of your offering. For example, Revenue Per Year (Sample App).
Folder	CMA Reports

### USER PERMISSIONS

To customize CMA reports:

- CMA Admin User

To create, edit, and delete reports:

- Create and Customize Reports  
AND  
Report Builder

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.

### [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.

#### 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.

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.

SEE ALSO:

[Modify a Notification Template in the Checkout Management App](#)


#### USER PERMISSIONS

To enable, disable, or customize notifications:

- CMA Admin User

## 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.

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.


### USER PERMISSIONS

To configure the Checkout Management App:

- CMA Admin User

## 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.

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**.

### USER PERMISSIONS

To configure the Checkout Management App:

- CMA Admin User

## 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.

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.

### USER PERMISSIONS

To manage apps:

- Customize Application

To view CMA logs:

- CMA Admin User



- c. Click **Checkout Logs**.

## CHAPTER 9    Monitor Performance with Analytics for AppExchange Partners

### In this chapter ...

- [Monitor Your AppExchange Performance with Marketplace Analytics](#)
- [AppExchange App Analytics](#)

Discover how customers find and interact with your AppExchange listing in the Marketplace Analytics dashboard. Learn how subscribers use your package by exploring App Analytics data.

# Monitor Your AppExchange Performance with Marketplace Analytics

---

Use Marketplace Analytics to discover how customers find and interact with your AppExchange listings.

## [Marketplace Analytics Dashboard](#)

The Marketplace Analytics dashboard uses activity metrics, trends, and visualizations to show how customers find and interact with your AppExchange listing.

## [Get Started with the Marketplace Analytics Dashboard](#)

View the Marketplace Analytics dashboard to see how your AppExchange listing is performing. To allow team members to view the dashboard, assign them permission in the Partner Community. To explore data outside of the dashboard, export it. To give feedback about the dashboard, use the Marketplace Analytics feedback tool.

## [Marketplace Analytics FAQs](#)

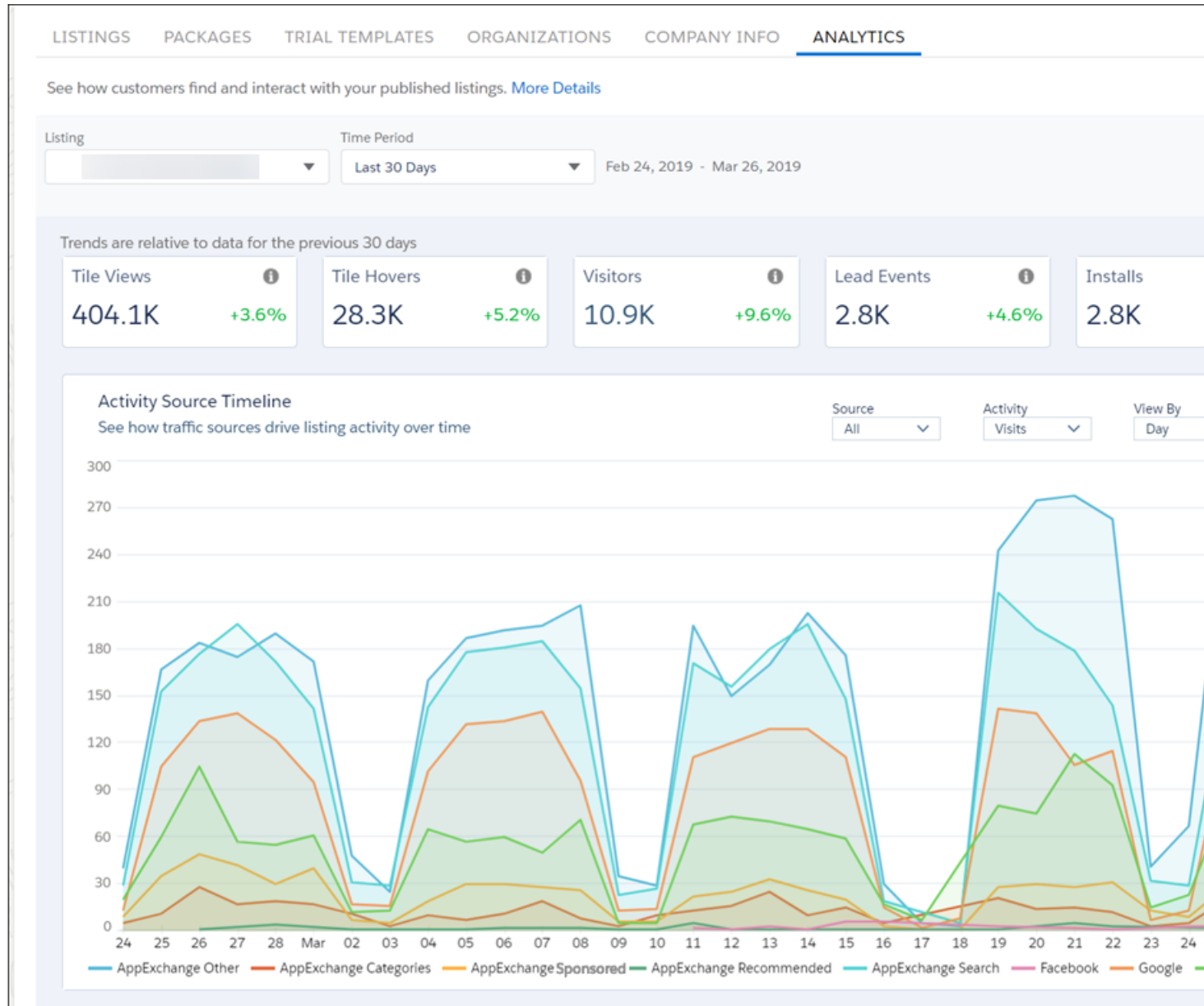
Here are some answers to frequently asked questions about Marketplace Analytics.

## Marketplace Analytics Dashboard

The Marketplace Analytics dashboard uses activity metrics, trends, and visualizations to show how customers find and interact with your AppExchange listing.



**Note:** Marketplace Analytics is available to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, visit <https://partners.salesforce.com>.



Check the dashboard to discover:

- How often customers view or hover over your listing
- How traffic sources, such as Google Ads, contribute to activity on your listing
- The AppExchange search terms that customers use to find your listing
- Which resources, such as screenshots, customers click or view as they explore your listing

Use these details to shape your AppExchange strategy and drive more visits, installs, and purchases.

#### [Activity Summary in the Marketplace Analytics Dashboard](#)

Use the activity summary to check your AppExchange listing's key metrics: tile views, tile hovers, visitors, lead events, and installs. Each activity metric includes a trend indicator to compare how your listing is performing relative to previous time periods.

Filters in the Marketplace Analytics Dashboard

Use filters to focus on the AppExchange listing data that interests you. Global filters, such as the time period, affect all components in the dashboard. Local filters, such as the source, affect individual visualizations.

Visualizations in the Marketplace Analytics Dashboard

Use visualizations to explore your AppExchange listing’s data. Activity Source Timeline and Customer Engagement show how and when customers interact with your listing and its resources. Activity Sources and Top AppExchange Searches show total activity over time. Activity Summary by Region shows how customers around the world and within the United States interact with your listing and its resources. Lead Events Timeline and Lead Events show the contribution of specific listing activities to overall lead events. Chat Engagement shows how customers interact with your AppExchange Chat implementation.

Marketplace Analytics CSV Files

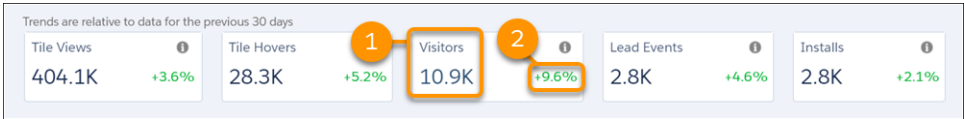
You can export data from the Marketplace Analytics dashboard in comma-separated value (.csv) format. When you export data, Marketplace Analytics creates a separate .csv file for each dashboard visualization and packages all the files in a .zip file.

What’s the Difference Between Lead Events and Leads in Marketplace Analytics?

Learn how we define lead events for your AppExchange listing and how they differ from the leads that appear in your Salesforce org.

Activity Summary in the Marketplace Analytics Dashboard

Use the activity summary to check your AppExchange listing’s key metrics: tile views, tile hovers, visitors, lead events, and installs. Each activity metric includes a trend indicator to compare how your listing is performing relative to previous time periods.



Element	Description
Activity Metric (1)	Number of times that an event or interaction occurred during a time period. For values over 1,000, the dashboard shows a rounded number. To view the exact number, hover over the metric.
Trend Indicator (2)	Percentage change relative to a previous time period. A positive value represents a period-over-period increase, and a negative value represents a period-over-period decrease.

The default time period is 30 days, but you can choose another fixed time period or define a custom one.

Example:



This summary of the example solution called Appy’s Maps shows that it received 98,000 tile views (3) in the past 7 days, which is a 3.6% increase (4) compared to the previous 7-day period (5).

## Filters in the Marketplace Analytics Dashboard


Use filters to focus on the AppExchange listing data that interests you. Global filters, such as the time period, affect all components in the dashboard. Local filters, such as the source, affect individual visualizations.

### Global Filters

Global Filter	Description
Listing (1)	Select an AppExchange listing to view. You can view only your company's published listings in the dashboard.
Time Period (2)	Select the time period that you want to explore. You can select one of the fixed time periods, such as the last quarter, or you can define a custom period. The time period's start and end dates appear next to the filter.

### Local Filters

Local Filter	Description	Visualization
Source (3)	Select the traffic sources to show in the visualization. Traffic sources help you understand where an activity on your AppExchange listing originated, such as an AppExchange search or a Facebook ad.	<ul style="list-style-type: none"> <li>Activity Source Timeline</li> <li>Activity Summary by Region</li> </ul>
Activity (4)	Select the activity metrics to show in the visualization. An activity metric tells you how often an event or interaction occurred on your AppExchange listing.	<ul style="list-style-type: none"> <li>Activity Source Timeline</li> <li>Activity Sources</li> <li>Activity Summary by Region</li> <li>Chat Engagement</li> <li>Customer Engagement</li> <li>Top AppExchange Searches</li> </ul>

Local Filter	Description	Visualization
View By (5)	<p>Adjust the time scale of the visualization, such as days, weeks, months, or quarters.</p> <p>In the x-axis of a visualization, weeks are formatted as <math>Wn</math>, where <math>n</math> is a week number. For example, <math>W1</math> represents the first week of the year. Likewise, quarters are formatted as <math>Qn</math>, where <math>n</math> is a quarter number. For example, <math>Q4</math> represents the fourth quarter of the year.</p> <p>For both weeks and quarters, the year starts on January 1.</p>	<ul style="list-style-type: none"><li>• Activity Source Timeline</li><li>• Chat Engagement</li><li>• Customer Engagement</li><li>• Lead Events Timeline</li></ul>
Lead Type	<p>Select the types of lead events to show in the visualization, such as lead events from demo views.</p> <p> <b>Note:</b> Marketplace Analytics categorizes lead events by listing activity starting on April 16, 2021. Before that date, we show only historical lead events.</p>	<ul style="list-style-type: none"><li>• Lead Events Timeline</li><li>• Lead Events</li></ul>

## Visualizations in the Marketplace Analytics Dashboard

Use visualizations to explore your AppExchange listing's data. Activity Source Timeline and Customer Engagement show how and when customers interact with your listing and its resources. Activity Sources and Top AppExchange Searches show total activity over time. Activity Summary by Region shows how customers around the world and within the United States interact with your listing and its resources. Lead Events Timeline and Lead Events show the contribution of specific listing activities to overall lead events. Chat Engagement shows how customers interact with your AppExchange Chat implementation.

### Activity Source Timeline

See how internal and external traffic sources contribute to activity on your AppExchange listing over a specific time period. Internal traffic originates on the AppExchange website, such as a customer who clicks a personalized recommendation to reach your listing. External traffic originates outside of AppExchange, such as a customer who clicks a Facebook ad to reach your listing.

### Activity Sources

See how traffic sources drive activity on your AppExchange listing. To see a breakdown of activities for a specific day, week, month, or quarter, use the Activity Source Timeline.

### Customer Engagement

See how customers interact with your listing and its resources over time. Resources include screenshots, demos, test drives, and other items that you've added to your listing in the Publishing Console.

### Top AppExchange Searches

See the 10 search terms that result in the most activity on your listing. Only searches performed with the search bar on the AppExchange website are included. Search terms from external search engines aren't available.

### Activity Summary by Region

See how internal and external traffic sources contribute to activity on your AppExchange listing around the world and within the United States. Internal traffic originates on the AppExchange website, such as a customer who clicks a personalized recommendation to reach your listing. External traffic originates outside of AppExchange, such as a customer who clicks a Facebook ad to reach your listing.

### Lead Events Timeline

See the activities that drive lead events on your AppExchange listing.

### Lead Events

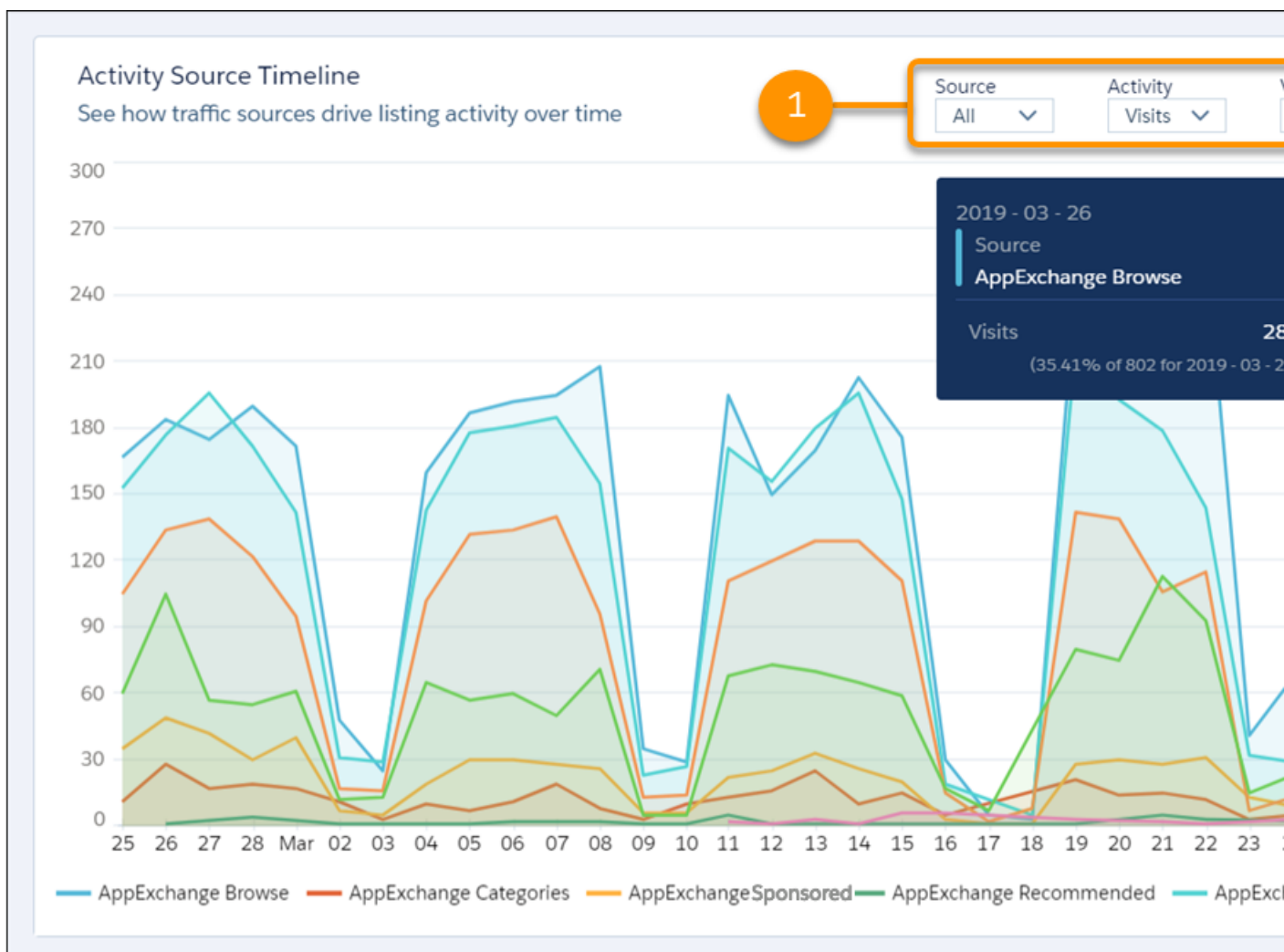
See the activities that drive lead events on your AppExchange listing. To see a breakdown of lead events over time, use the Lead Events Timeline.

### Chat Engagement

See how customers interact with your AppExchange Chat experiences, such as the number of conversations that your sales reps hosted.

## Activity Source Timeline

See how internal and external traffic sources contribute to activity on your AppExchange listing over a specific time period. Internal traffic originates on the AppExchange website, such as a customer who clicks a personalized recommendation to reach your listing. External traffic originates outside of AppExchange, such as a customer who clicks a Facebook ad to reach your listing.



To change traffic sources, activities, or time scale, adjust the local filters (1). The y-axis resizes based on the traffic sources and activities that you select. To resize the x-axis, change the View By filter. To see exact values, hover over a line in the chart (2).

If the visualization doesn't display data, try filtering by different metrics, or change the time period.

## Definitions

Here's how we define the metrics that appear in this visualization.

Metric	Description
Installs	Installs of your solution initiated on AppExchange, your website, or from a code repository. For AppExchange installs, we count the number of successful completions of the Get It Now installation flow. Includes installs in production and sandbox orgs.
Lead Events	Lead events on your listing. Events include: demos, test drives, chat interactions, Learn More clicks, and Get It Now clicks or installs. A customer who clicks Get It Now and then installs your solution is counted as a single lead event.
Tile Hovers	Hovers over your listing tile. To qualify as a hover, the customer must pause long enough over the tile to display the listing detail popover. The count includes repeat hovers by the customer.
Tile Views	Views of your listing tile. To qualify as a view, the entire tile must be visible in the customer's browser. Includes any repeat views by the customer.
Visits	Visits to your listing. Includes repeat visits by the customer.

These internal traffic sources are associated with activities.

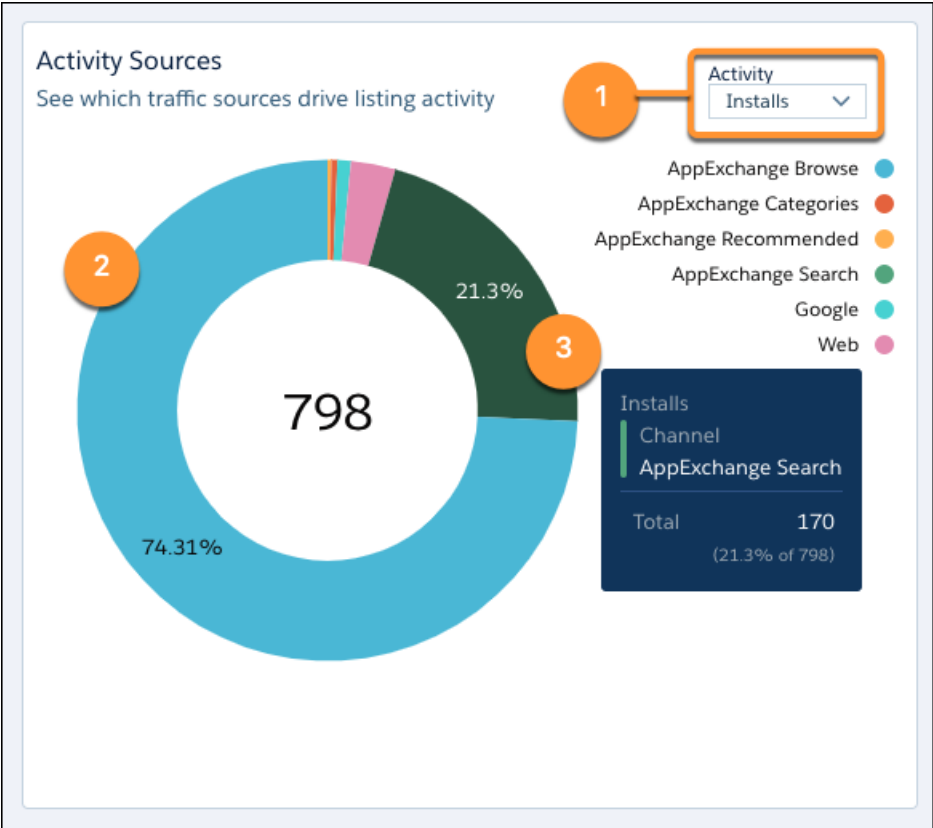
Traffic Source	Description
AppExchange Browse	Activity by customers who reached your listing from areas of AppExchange that aren't included in other sources. For example, a customer who browses a Product Collection, an Industry Collection, or the AppExchange home page.
AppExchange Categories	Activity by customers who reached your listing from one of AppExchange's Solutions by Type categories.
AppExchange Sponsored	Activity by customers who reached your listing from AppExchange's Sponsored Solutions section.
AppExchange Recommended	Activity by customers who reached your listing from an AppExchange personalized recommendation. Includes Recommended for You and Appy's Picks for You.
AppExchange Search	Activity by customers who reached your listing from a search made using the AppExchange search bar.

These external traffic sources are associated with activities.

Traffic Source	Description
Facebook	Activity by customers who reached your listing from a Facebook page or ad. Includes organic traffic and traffic from ads shown on the Facebook site or Facebook's Audience Network.
Google	Activity by customers who reached your listing from a Google search or ad. Includes organic search traffic and traffic from ads shown on the Google Search Network or Google Display Network.
Web	Activity by customers who reached your listing from a web source that isn't affiliated with Facebook or Google. Includes traffic from your company's website.

Activity Sources

See how traffic sources drive activity on your AppExchange listing. To see a breakdown of activities for a specific day, week, month, or quarter, use the Activity Source Timeline.



To change activities, adjust the local filter (1). The percentage (2) within a chart segment represents the contribution of the traffic source to the activity’s total. To see exact values, hover over a chart segment (3).

If the visualization doesn’t display data, try filtering by different metrics, or change the time period.

Definitions

Here’s how we define the metrics that appear in this visualization.

Metric	Description
Demos	Demo button clicks associated with the search term.
Installs	Installs associated with the search term.  Qualifying installs include those initiated on AppExchange, your website, or from a code repository. For AppExchange installs, the number represents successful completions of the Get It Now installation flow, and includes installs in production and sandbox orgs.

Metric	Description
Lead Events	Lead events associated with the search term.  Lead events include: demos, test drives, chat interactions, Learn More clicks, and Get It Now clicks or installs. A customer who clicks Get It Now and then installs your solution is counted as a single lead event.
Test Drives	Test drive button clicks associated with the search term.
Visits	Unique listing visits associated with the search term.

These internal traffic sources are associated with activities.

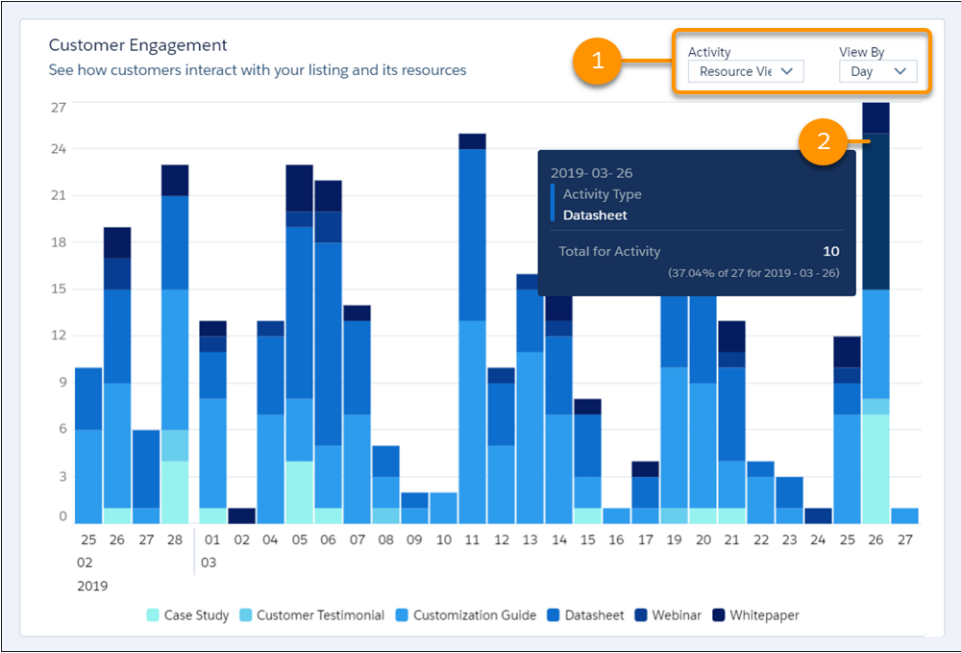
Traffic Source	Description
AppExchange Browse	Activity by customers who reached your listing from areas of AppExchange that aren't included in other sources. For example, a customer who browses a Product Collection, an Industry Collection, or the AppExchange home page.
AppExchange Categories	Activity by customers who reached your listing from one of AppExchange's Solutions by Type categories.
AppExchange Sponsored	Activity by customers who reached your listing from AppExchange's Sponsored Solutions section.
AppExchange Recommended	Activity by customers who reached your listing from an AppExchange personalized recommendation. Includes Recommended for You and Appy's Picks for You.
AppExchange Search	Activity by customers who reached your listing from a search made using the AppExchange search bar.

These external traffic sources are associated with activities.

Traffic Source	Description
Facebook	Activity by customers who reached your listing from a Facebook page or ad. Includes organic traffic and traffic from ads shown on the Facebook site or Facebook's Audience Network.
Google	Activity by customers who reached your listing from a Google search or ad. Includes organic search traffic and traffic from ads shown on the Google Search Network or Google Display Network.
Web	Activity by customers who reached your listing from any web source that isn't affiliated with Facebook or Google. Includes traffic from your company's website.

## Customer Engagement

See how customers interact with your listing and its resources over time. Resources include screenshots, demos, test drives, and other items that you've added to your listing in the Publishing Console.



To change activities or the time scale, adjust the local filters (1). The y-axis resizes based on the activities that you select. To resize the x-axis, change the View By filter. To see exact values, hover over a chart segment (2).

If the visualization doesn't display data, try filtering by different metrics, or change the time period.

Definitions

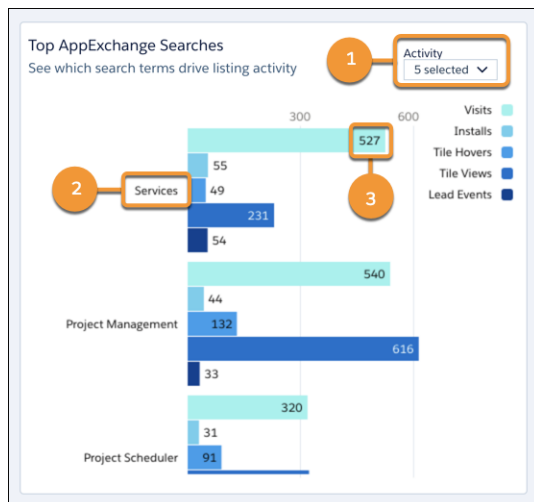
Here's how we define the metrics that appear in this visualization.

Metric	Description
Case Studies	Views of your listing's case studies.
Customization Guides	Views of your listing's customization guides.
Data Sheets	Views of your listing's data sheets.
Demos	Clicks on your listing's Watch Demo button.
Get It Nows	Clicks on your listing's Get It Now button. Customers who click the button start the Get It Now installation flow, but might not complete it.
Tile Hovers	Hovers over your listing tile. To qualify as a hover, the customer must pause long enough over the tile to display the listing detail popover. Includes repeat hovers by the customer.
Installs	Installs of your solution initiated on AppExchange, your website, or from a code repository. For AppExchange installs, we count the number of successful completions of the Get It Now installation flow. Includes installs in production and sandbox orgs.
Lead Events	Lead events on your listing. Events include: demos, test drives, chat interactions, Learn More clicks, and Get It Now clicks or installs. A customer who clicks Get It Now and then installs your solution is counted as a single lead event.

Metric	Description
Saves	Clicks on your listing's Save button.
Screenshot <i>n</i>	Views of screenshot number <i>n</i> . This number corresponds to the number shown in the image carousel on your listing.
Test Drives	Clicks on your listing's Test Drive button.
Testimonials	Views of your listing's testimonials.
Tile Views	Views of your listing tile. To qualify as a view, the entire tile must be visible in the customer's browser. Includes repeat views by the customer.
Webinars	Views of your listing's webinars.
White Papers	Views of your listing's white papers.
Visits	Visits to your listing. Includes repeat visits by the customer.

## Top AppExchange Searches

See the 10 search terms that result in the most activity on your listing. Only searches performed with the search bar on the AppExchange website are included. Search terms from external search engines aren't available.



Select the activities that you want to view (1). The search term (2) associated with the activities appears on the left of the chart. Activity metric values appear on the right (3).

Depending on your filter selections, some search terms might not be visible. To see all available search terms, position your pointer over the visualization and scroll down.

If the visualization doesn't display data, try filtering by different metrics, or change the time period.

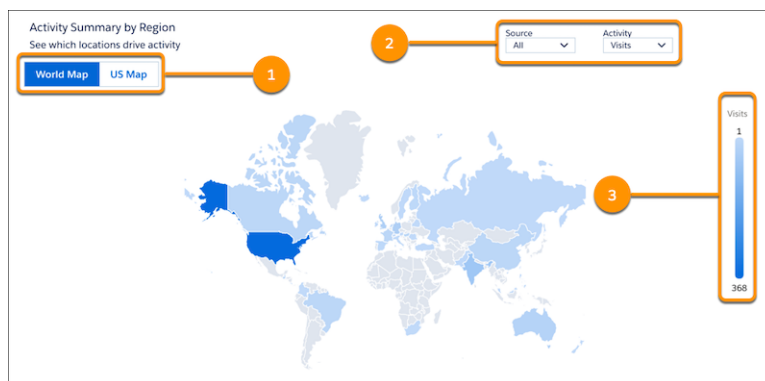
## Definitions

Here's how we define the metrics that appear in this visualization.

Metric	Description
Installs	Installs of your solution initiated on AppExchange, your website, or from a code repository. For AppExchange installs, we count the number of successful completions of the Get It Now installation flow. Includes installs in production and sandbox orgs.
Lead Events	Lead events on your listing. Events include: demos, test drives, chat interactions, Learn More clicks, and Get It Now clicks or installs. A customer who clicks Get It Now and installs your solution is counted as a single lead event.
Tile Hovers	Hovers over your listing tile. To qualify as a hover, the customer must pause long enough over the tile to display the listing detail popover. Includes any repeat hovers by the customer.
Tile Views	Views of your listing tile. To qualify as a view, the entire tile must be visible in the customer's browser. Includes any repeat views by the customer.
Visitors	Unique listing visitors. If a customer visits your listing more than once in a 30-day period, only a single visitor is counted.

## Activity Summary by Region

See how internal and external traffic sources contribute to activity on your AppExchange listing around the world and within the United States. Internal traffic originates on the AppExchange website, such as a customer who clicks a personalized recommendation to reach your listing. External traffic originates outside of AppExchange, such as a customer who clicks a Facebook ad to reach your listing.



View data by country or by US state or territory. To view data by country, select the World Map; to view data by US state or territory, select the US Map (1). To change traffic sources or activities, adjust the local filters (2). Notice that the gradient displays the range of values for the activity you select (3). To see detailed metrics for a selected area on the World Map, hover over a country. Or on the US Map, hover over a state or territory.

If the visualization doesn't display data, try filtering by different metrics.

## Definitions

Here's how we define the metrics that appear in this visualization.

Metric	Description
Installs	Installs of your solution initiated on AppExchange, your website, or from a code repository. For AppExchange installs, we count the number of successful completions of the Get It Now installation flow. Includes installs in production and sandbox orgs.
Lead Events	Lead events on your listing. Events include: demos, test drives, chat interactions, Learn More clicks, and Get It Now clicks or installs. A customer who clicks Get It Now and then installs your solution is counted as a single lead event.
Tile Hovers	Hovers over your listing tile. To qualify as a hover, the customer must pause long enough over the tile to display the listing detail popover. The count includes repeat hovers by the customer.
Tile Views	Views of your listing tile. To qualify as a view, the entire tile must be visible in the customer's browser. Includes any repeat views by the customer.
Visits	Visits to your listing. Includes repeat visits by the customer.

These internal traffic sources are associated with activities.

Traffic Source	Description
AppExchange Browse	Activity by customers who reached your listing from areas of AppExchange that aren't included in other sources. For example, a customer who browses a Product Collection, an Industry Collection, or the AppExchange home page.
AppExchange Categories	Activity by customers who reached your listing from one of AppExchange's Solutions by Type categories.
AppExchange Sponsored	Activity by customers who reached your listing from AppExchange's Sponsored Solutions section.
AppExchange Recommended	Activity by customers who reached your listing from an AppExchange personalized recommendation. Includes Recommended for You and Appy's Picks for You.
AppExchange Search	Activity by customers who reached your listing from a search made using the AppExchange search bar.

These external traffic sources are associated with activities.

Traffic Source	Description
Facebook	Activity by customers who reached your listing from a Facebook page or ad. Includes organic traffic and traffic from ads shown on the Facebook site or Facebook's Audience Network.
Google	Activity by customers who reached your listing from a Google search or ad. Includes organic search traffic and traffic from ads shown on the Google Search Network or Google Display Network.
Web	Activity by customers who reached your listing from a web source that isn't affiliated with Facebook or Google. Includes traffic from your company's website.

Lead Events Timeline

See the activities that drive lead events on your AppExchange listing.




To change lead type or the time scale, adjust the local filters (1). The y-axis resizes based on the lead type that you select. To resize the x-axis, change the View By filter. To see exact values, hover over a line in the chart (2).

If the visualization doesn't display data, try filtering by different metrics, or change the time period.

Definitions

Here's how we define the metrics that appear in this visualization.

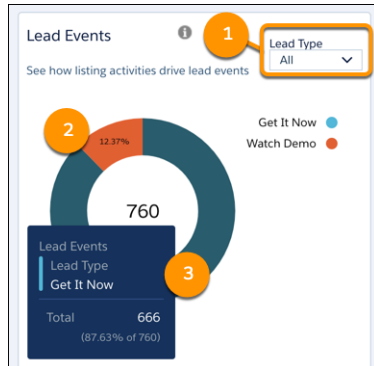
Metric	Description
Chat	A lead event that results from an AppExchange Chat interaction. These interactions include conversations with a human or chatbot and meetings booked.   <b>Note:</b> AppExchange Chat is required to view chat data in Marketplace Analytics. This feature is available to eligible Salesforce partners through the AppExchange Marketing Program (AMP). Learn more about AppExchange Chat in the <a href="#">Salesforce Partner Community</a> .
Demo	A lead event that results from a Watch Demo button click.
Get It Now	A lead event that results from a Get It Now button click.
Learn More	A lead event that results from a Learn More button click.
Historical	A lead event that occurred on your listing before April 16, 2021. Historical lead events are created by test drives, demos, Learn More clicks, and installs or Get It Now clicks, but aren't categorized.
Test Drive	A lead event that results from a Test Drive button click.

Considerations

Marketplace Analytics categorizes lead events by listing activity starting on April 16, 2021. Before that date, we show only *historical* lead events.

## Lead Events


See the activities that drive lead events on your AppExchange listing. To see a breakdown of lead events over time, use the Lead Events Timeline.



To change lead types, adjust the local filter (1). The percentage (2) within a chart segment represents the contribution of the lead type to the total number of lead events. To see exact values, hover over a chart segment (3).

## Definitions

Here's how we define the metrics that appear in this visualization.

Metric	Description
Chat	A lead event that results from an AppExchange Chat interaction. These interactions include customer conversations with a human or chatbot and meetings booked.   <b>Note:</b> AppExchange Chat is required to view chat data in Marketplace Analytics. This feature is available to eligible Salesforce partners through the AppExchange Marketing Program (AMP). Learn more about AppExchange Chat in the <a href="#">Salesforce Partner Community</a> .
Demo	A lead event that results from a Watch Demo button click.
Get It Now	A lead event that results from a Get It Now button click.
Learn More	A lead event that results from a Learn More button click.
Historical	A lead event that occurred on your listing before April 16, 2021. Historical lead events are created by test drives, demos, Learn More clicks, and installs or Get It Now clicks, but aren't categorized.
Test Drive	A lead event that results from a Test Drive button click.

## Considerations

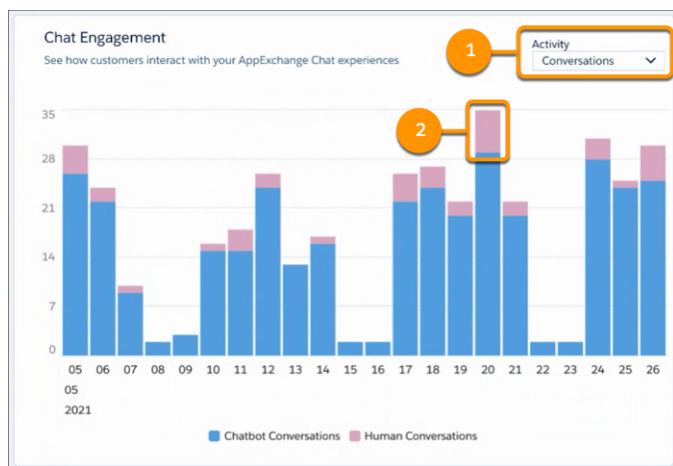
Marketplace Analytics categorizes lead events by listing activity starting on April 16, 2021. Before that date, we show only *historical* lead events.

## Chat Engagement

See how customers interact with your AppExchange Chat experiences, such as the number of conversations that your sales reps hosted.



**Note:** AppExchange Chat is required to view chat data in Marketplace Analytics. This feature is available to eligible Salesforce partners through the AppExchange Marketing Program (AMP). Learn more about AppExchange Chat in the [Salesforce Partner Community](#).



To change activities, adjust the local filter (1). To see exact values, hover over a chart segment (2).

If the visualization doesn't display data, first verify that AppExchange Chat is enabled on your listing. Then try filtering by different metrics, or change the time period.

## Definitions

Here's how we define the metrics that appear in this visualization.

Metric	Description
Chat Leads Created	Unique leads passed from AppExchange Chat to your CRM implementation, such as Salesforce or Pardot. A single chat lead can be associated with multiple chat lead events.  For example, if a customer chats with your reps several times across multiple listing visits, we record a lead event for each interaction. However, we pass only one chat lead to your CRM. This behavior prevents unwanted duplication of lead records in your CRM.
Chat Lead Event	Lead events on your listing from AppExchange Chat activity, such as human or chatbot conversations.
Chatbot Conversation	Conversations between a customer and a chatbot experience that you configure.
Human Conversation	Real-time conversations between a customer and a rep at your company.
Meetings Booked	Meetings booked with a customer during a live chat or chatbot conversation.

## Marketplace Analytics CSV Files

You can export data from the Marketplace Analytics dashboard in comma-separated value (.csv) format. When you export data, Marketplace Analytics creates a separate .csv file for each dashboard visualization and packages all the files in a .zip file.


We format .csv files as follows.

- The first row is the header and provides column names. Subsequent rows represent records.
- Within rows, values are separated by commas.
- Negative values are prefixed with a minus sign.

 **Note:** Activity Summary by Region data isn't available in .csv format.

## Activity Source Timeline File

Provides data from the Activity Source Timeline visualization with your global and local filter selections applied.


 **Example:** This example shows the header row and four rows of sample data. The following filters were applied.

- Global filter set to show data for the last 30 days.
- Local filters set to show visits by day for these traffic sources: AppExchange Featured and AppExchange Search.

```
Date,Source,Activity,Count of Activity
2019-01-01,AppExchange Featured,Visits,25
2019-01-01,AppExchange Search,Visits,50
2019-01-02,AppExchange Featured,Visits,30
2019-01-02,AppExchange Search,Visits,60
```

## Customer Engagement File

Provides data from the Customer Engagement visualization with your global and local filter selections applied.


 **Example:** This example shows the header row and 4 rows of sample data. The following filters were applied.

- Global filter set to show data for the last 30 days.
- Local filters set to show resource views by day.

```
Date,Activity,Count of Activity
2019-01-01,Customization Guide,10
2019-01-01,Datasheet,20
2019-01-02,Customization Guide,20
2019-01-02,Datasheet,40
```

## Activity Sources File


Provides data from the Activity Sources visualization with your global and local filter selections applied.

 **Example:** This example shows the header row and four rows of sample data. The following filters were applied.

- Global filter set to show data for the last 30 days.
- Local filter set to show visits.


```
Source,Activity,Count of Activity,Percentage of Total Activity,Rank
AppExchange Browse,Visits,500,20.41,1
```

```
AppExchange Categories,Visits,450,18.37,2
AppExchange Search,Visits,400,16.33,3
AppExchange Recommended,350,14.29,4
```

 **Note:** For brevity, this sample shows only four traffic sources: AppExchange Browse, AppExchange Categories, AppExchange Search, and AppExchange Recommended. The file that you export from your dashboard provides all traffic sources.

## Top AppExchange Searches File

Provides data from the Top AppExchange Searches visualization with your global and local filter selections applied.


 **Example:** This example shows the header row and four rows of sample data. The following filters were applied.

- Global filter set to show data for the last 30 days.
- Local filters set to show the top search terms associated with visits and demos.

```
Search Term,Activity,Count of Activity
Geolocation,Visits,50
Geolocation,Demos,40
Maps,Visits,30
Maps,Demos,20
```

## Lead Events Timeline File

Provides data from the Lead Events Timeline visualization with your global and local filter selections applied.

 **Example:** This example shows the header row and four rows of sample data. The following filters were applied.

- Global filter set to show data for the last 30 days.
- Local filter set to show Get It Now clicks and demos.

```
Date,Lead Type,Count of Leads
2021-05-03,Get It Now,31
2021-05-03,Watch Demo,3
2021-05-04,Get It Now,40
2021-05-04,Watch Demo,8
```

## Lead Events File


Provides data from the Lead Events visualization with your global and local filter selections applied.

 **Example:** This example shows the header row and two rows of sample data. The following filters were applied.


- Global filter set to show data for the last 30 days.
- Local filter set to show Get It Now clicks and demos.

```
Lead Type,Count of Leads,Percentage of Total Leads
Get It Now,666,87.6
Watch Demo,94,12.4
```

## Chat Engagement File

 **Note:** AppExchange Chat is required to view chat data in Marketplace Analytics. This feature is available to eligible Salesforce partners through the AppExchange Marketing Program (AMP). Learn more about AppExchange Chat in the [Salesforce Partner Community](#).

Provides data from the Chat Engagement visualization with your global and local filter selections applied.

 **Example:** This example shows the header row and four rows of sample data. The following filters were applied.

- Global filter set to show data for the last 30 days.
- Local filter set to show conversations.


```
Date,Activity,Type,Total for Activity
2021-05-03,Conversations,Chatbot Conversations,26
2021-05-03,Conversations,Human Conversations,4
2021-05-04,Conversations,Chatbot Conversations,22
2021-05-04,Conversations,Human Conversations,2
```

## What's the Difference Between Lead Events and Leads in Marketplace Analytics?

Learn how we define lead events for your AppExchange listing and how they differ from the leads that appear in your Salesforce org.

Marketplace Analytics records a lead event when a customer visits your listing and:

- Watches a demo
- Takes a test drive
- Interacts with AppExchange Chat
- Clicks **Get It Now**
- Clicks **Learn More** (applies only to consultant listings)
- Installs your solution

 **Note:** AppExchange Chat is required to view chat data in Marketplace Analytics. This feature is available to eligible Salesforce partners through the AppExchange Marketing Program (AMP). Learn more about AppExchange Chat in the [Salesforce Partner Community](#).

If you configured Web-to-Lead and enabled lead collection for the listing, each of these activities also creates a lead in your org. However, custom lead routing rules, customer contact preferences, and Web-to-Lead reCaptcha can cause the number of leads in your org to differ from the number of lead events shown in Marketplace Analytics.

## Custom Lead Routing Rules

Typically, you set up custom lead routing rules to prevent duplicate or unwanted leads from reaching your sales team. Here are some common examples of routing rules where Marketplace Analytics lead events aren't recorded as leads in your org.

Lead Routing Rule	Example	Marketplace Analytics	Your Org
<b>Domain Restriction</b> You filter leads from customers whose email address includes your company's domain.	An employee at your company watches your listing's demo video and uses a company email address when	A lead event is recorded.	A lead isn't recorded. The lead routing rule filters out the lead.

Lead Routing Rule	Example	Marketplace Analytics	Your Org
	<p>AppExchange asks for contact information.</p> <p>In this scenario, Marketplace Analytics records a lead event, but the lead routing rule filters the lead in your org.</p>		
<p><b>Duplicate Email Addresses</b></p> <p>You filter leads associated with an email address that's been captured in an existing lead.</p>	<p>A new customer goes to your listing and watches a video, takes a test drive, and installs your solution. For each activity, the customer provides the same email address.</p> <p>In this scenario, Marketplace Analytics records three lead events: one for each activity. In your org, the lead routing rule creates a lead for the first activity. The others are marked as duplicates because they're associated with the same email address.</p>	Three lead events are recorded, one for each activity.	A lead is created for the first activity only. The others are marked as duplicates because they're associated with the same email address.

## Trailblazer.me Contact Preferences

In a customer's Trailblazer.me settings, the customer can choose to share their contact info with, and allow contact from, AppExchange providers. Their choices impact lead creation in your org. For customers who allow provider contact, AppExchange lead events are recorded in Marketplace Analytics and propagate to your org as leads. Here are common examples of how contact preferences impact lead creation.

Trailblazer.me Contact Preference	Example	Marketplace Analytics	Your Org
Allow	A prospect who allows provider contact watches your listing's demo video.	A lead event is recorded.	If custom lead routing rules don't filter out the lead, then a lead is created in your org.
Prohibit	A prospect who prohibits provider contact takes a test drive of your solution.	A lead event is recorded.	If custom lead routing rules don't filter out the lead, then a lead is created in your org. The lead is flagged as contact prohibited.

Customers can override their default Trailblazer.me contact preferences when interacting with AppExchange listings. AppExchange recognizes when a customer interacts with your listing in a way that you chose to collect leads for. These customers are prompted to fill out the AppExchange lead sign-up form.

The screenshot shows a 'Watch Demo' form. At the top, it says 'Before we continue, the provider requests your contact information.' Below this, there's a section titled 'Here are the details we'll share from your profile' with an 'Edit Profile' link. The details include: First Name (Appy), Last Name (Force), Job Title (Developer), Email (appy@example.com), Company (Salesforce), Country (United States), and State/Province (Massachusetts). There's a checkbox for 'I have read and agree to the terms and conditions.' Below that, a note states: 'By submitting this request, you agree to share your information with Salesforce and the provider of this listing, Jitterbit, Inc.' The listing is identified as 'Labs Dashboard'. There's another checkbox for 'Allow the provider to contact me by email, phone, or SMS about other products or services I might like'. At the bottom, there's a checkbox for 'Ask me about sharing my contact information every time I take a Test Drive or watch a demo' with a note: 'You can update your preferences later in your Salesforce community profile.' There are 'Cancel' and 'Submit' buttons at the bottom right.

The form prepopulates with the customer's contact info and preferences from their Trailblazer.me settings. On the form, the customer can choose to allow or prohibit provider contact, effectively overriding the contact preference that they set in their Trailblazer.me profile.

## Web-to-Lead reCaptcha Verification

To receive AppExchange leads, disable Require reCaptcha Verification in your org's Web-to-Lead settings.

SEE ALSO:

[Collect AppExchange Leads](#)

[Troubleshoot AppExchange Leads](#)

## Get Started with the Marketplace Analytics Dashboard

View the Marketplace Analytics dashboard to see how your AppExchange listing is performing. To allow team members to view the dashboard, assign them permission in the Partner Community. To explore data outside of the dashboard, export it. To give feedback about the dashboard, use the Marketplace Analytics feedback tool.

[Grant Access to the Marketplace Analytics Dashboard](#)

The Manage Listings permission provides access to the Marketplace Analytics dashboard. Assign this permission to the people on your team who monitor the performance of your AppExchange solution.

[View an AppExchange Listing in the Marketplace Analytics Dashboard](#)

You can check how an AppExchange listing is performing in the Marketplace Analytics dashboard.

[View the Marketplace Analytics Glossary](#)

To see definitions for activity metrics and traffic sources, open the Marketplace Analytics glossary.

[Export Data from the Marketplace Analytics Dashboard](#)


To explore Marketplace Analytics data outside of the dashboard, export it. Data is exported in comma-separated value (.csv) format with your global and local filter selections applied.

[Give Feedback About Marketplace Analytics](#)

To give feedback about Marketplace Analytics, use the dashboard's feedback tool. Tell us what's working well, what we can improve, or anything else you'd like to share about your experience.

## Grant Access to the Marketplace Analytics Dashboard

The Manage Listings permission provides access to the Marketplace Analytics dashboard. Assign this permission to the people on your team who monitor the performance of your AppExchange solution.

 **Note:** The Manage Listings permission provides access to all Publishing Console features, including the ability to create, edit, and publish listings. We suggest assigning the permission to the people on your team who also manage your company's AppExchange listing.

1. Log in to the [Salesforce Partner Community](#).
2. Click **Manage Users**.
3. Search for a user at your company.
4. Under Listings, select the checkbox.

### USER PERMISSIONS

To assign permissions to Partner Community users:

- Manage Users

## View an AppExchange Listing in the Marketplace Analytics Dashboard

You can check how an AppExchange listing is performing in the Marketplace Analytics dashboard.

1. Log in to the [Salesforce Partner Community](#).
2. Click **Publishing**.
3. Click **Analytics**.
4. Select the listing to view.


### USER PERMISSIONS

To view Marketplace Analytics:

- Manage Listings

## View the Marketplace Analytics Glossary

To see definitions for activity metrics and traffic sources, open the Marketplace Analytics glossary.

1. Log in to the [Salesforce Partner Community](#).
2. Click **Publishing**.
3. Click **Analytics**.
4. Click .


### USER PERMISSIONS

To view Marketplace Analytics:

- Manage Listings

## Export Data from the Marketplace Analytics Dashboard

To explore Marketplace Analytics data outside of the dashboard, export it. Data is exported in comma-separated value (.csv) format with your global and local filter selections applied.

1. Log in to the [Salesforce Partner Community](#).
2. Click **Publishing**.
3. Click **Analytics**.
4. Select a listing, and choose the time period.
5. For each visualization, select activity metrics and, if available, change the time scale.
6. Click , and then click **Export**.

### USER PERMISSIONS

To view Marketplace Analytics:

- Manage Listings

## Give Feedback About Marketplace Analytics

To give feedback about Marketplace Analytics, use the dashboard's feedback tool. Tell us what's working well, what we can improve, or anything else you'd like to share about your experience.

1. Log in to the [Salesforce Partner Community](#).
2. Click **Publishing**.
3. Click **Analytics**.
4. Click **Feedback**.
5. Share your feedback about the dashboard, and then click **Give Feedback**.

### USER PERMISSIONS

To view Marketplace Analytics:

- Manage Listings

## Marketplace Analytics FAQs

Here are some answers to frequently asked questions about Marketplace Analytics.

[Can I grant access to the Marketplace Analytics dashboard but not other publishing features?](#)

[What's the earliest date that Marketplace Analytics data is available for?](#)

[Is aggregate data for all AppExchange listings available in Marketplace Analytics?](#)

[Is there a Marketplace Analytics API?](#)

[Why doesn't data appear in my Marketplace Analytics activity summary or visualization?](#)

[Can I view my consulting service listing in the Marketplace Analytics dashboard?](#)

[How often is Marketplace Analytics data updated?](#)

[Can I switch to the classic version of the Marketplace Analytics dashboard?](#)

[Can I customize Marketplace Analytics visualizations?](#)

[Can I import data into the Marketplace Analytics dashboard?](#)

[Does Marketplace Analytics change how customers experience my listing?](#)

[Why doesn't the sum of installs, demos, and test drives match the number of leads in Marketplace Analytics?](#)

[What's the difference between a Get It Now click and an install?](#)

### Can I grant access to the Marketplace Analytics dashboard but not other publishing features?

No. The Manage Listings permission provides access to all features in the Publishing Console. We suggest assigning this permission to the people on your team who also manage your company's AppExchange listing.

### What's the earliest date that Marketplace Analytics data is available for?

Marketplace Analytics data dates back to August 2019.

### Is aggregate data for all AppExchange listings available in Marketplace Analytics?

No. You can view only the data that's associated with your published AppExchange listings.

## Is there a Marketplace Analytics API?

No. However, you can export data from the Marketplace Analytics dashboard. To export data, go to the dashboard and click .

## Why doesn't data appear in my Marketplace Analytics activity summary or visualization?

Typically, this happens when Marketplace Analytics can't find data for the selected time period or activity metric. Try filtering by different metrics, or change the time period.

## Can I view my consulting service listing in the Marketplace Analytics dashboard?

Yes. Marketplace Analytics supports all listing types, including consulting service listings. If your listing doesn't have a managed package, some activity metrics, such as installs, won't have data.

## How often is Marketplace Analytics data updated?

Marketplace Analytics data is updated once per day.


## Can I switch to the classic version of the Marketplace Analytics dashboard?

No. We've retired the classic version of the dashboard. To share feedback with us about the latest version of the dashboard, use the Marketplace Analytics feedback tool.

## Can I customize Marketplace Analytics visualizations?

From the global filter menu, you can adjust the time period shown in visualizations. Within a visualization, you can choose the activity metrics that display and, for certain visualizations, time period scale. You can't modify the layout of the dashboard or change the appearance of individual visualizations.

## Can I import data into the Marketplace Analytics dashboard?

No. To use Marketplace Analytics data with an external dataset, use the dashboard's export tool. To export your data, go to the dashboard and click .

## Does Marketplace Analytics change how customers experience my listing?

No. Marketplace Analytics doesn't affect how customers browse, search for, or interact with listings on AppExchange.

## Why doesn't the sum of installs, demos, and test drives match the number of leads in Marketplace Analytics?

Typically, this happens when Web-to-Lead isn't set up in your org or when Web-to-Lead isn't configured correctly. To learn more about Web-to-Lead, search for "Generate Leads from Your Website for Your Sales Teams" in [Salesforce Help](#).

## What's the difference between a Get It Now click and an install?

The AppExchange installation process has several steps. To start the installation process, a customer clicks **Get It Now** on a listing. Marketplace Analytics records this interaction as a Get It Now click. Next, the customer chooses a destination for the package and agrees to our terms and conditions. Then, the customer clicks **Confirm and Install**. Marketplace Analytics records this interaction as an install.

## AppExchange App Analytics

---

AppExchange App Analytics provides usage data about how subscribers interact with your AppExchange solutions. You can use these details to identify attrition risks, inform feature development decisions, and improve user experience.



**Note:** AppExchange App Analytics is subject to certain usage restrictions as described in the [AppExchange Program Policies](#).

App Analytics is available for packages that have passed security review and are registered to a License Management App. Usage data is provided as log files, month-based usage summaries, or subscriber snapshots. All usage data is available as downloadable comma-separated value (.csv) files. To view the data in dashboard or visualization format, use [Tableau CRM](#) or a third-party analytics tool.

In a 24-hour period, you can download a maximum 20 GB of AppExchange App Analytics data.

### [Request AppExchange App Analytics](#)

To request access to AppExchange App Analytics package usage logs and subscriber snapshots, log a case in the Salesforce Partner Community. Package usage summaries are available on your package by default.

### [Download Package Log Files, Usage Summaries, and Subscriber Snapshots](#)

To request package log files, monthly usage summaries, and subscriber snapshots, use AppAnalyticsQueryRequest in SOAP API. Log files, usage summaries, and subscriber snapshots are downloadable comma-separated value (.csv) files.

### [AppExchange App Analytics Best Practices](#)

To plan and maximize your AppExchange App Analytics query strategy, follow our best practices. First, use file compression to reduce your data results file size. Second, schedule and automate your regular App Analytics queries. Third, plan, schedule, and automate catch-up queries to supplement your regular query data.

### [Package Usage Summaries](#)

Package usage summaries provide high-level metrics by calendar month. Discover how many users access your package and which operations they perform.

### [Package Log Files](#)

Analyze adoption and user behavior, then make informed feature development decisions based on data from package log files. AppExchange App Analytics tracks UI, API-based, Lightning-based, and Apex operations, and it logs each CRUD operation on components and custom objects in packages. Events from sandbox and trial orgs are tracked in package log files. Events from scratch orgs aren't tracked.

### [Subscriber Snapshots](#)

Subscriber snapshots provide a point-in-time summary of your subscribers' activity. Use subscriber snapshots to see usage trends by org and package over time.

### [Test Custom Integrations](#)

To test your custom integrations in a nonproduction environment, use AppExchange App Analytics Simulation Mode. Submit an App Analytics query request and receive sample usage data.

## Request AppExchange App Analytics

To request access to AppExchange App Analytics package usage logs and subscriber snapshots, log a case in the Salesforce Partner Community. Package usage summaries are available on your package by default.

 **Note:** AppExchange App Analytics is subject to certain usage restrictions as described in the [AppExchange Program Policies](#).

1. Log in to the [Salesforce Partner Community](#).
2. Click **Support**.
3. Click **New Case**.
4. Select **Other AppExchange Topics > Create a Case**.
5. Select **Enable App Analytics**.
6. Specify the package IDs that you want to track analytics data for.

Use the subscriber package ID that begins with 033. To retrieve a list of your second-generation managed package IDs, run `sfdx force:package:list --verbose` in Salesforce CLI.

We review your case, and notify you when App Analytics is enabled.

## Download Package Log Files, Usage Summaries, and Subscriber Snapshots


To request package log files, monthly usage summaries, and subscriber snapshots, use AppAnalyticsQueryRequest in SOAP API. Log files, usage summaries, and subscriber snapshots are downloadable comma-separated value (.csv) files.

To request access to AppExchange App Analytics, [log a case](#) in the Salesforce Partner Community.

Then determine which team members need CRUD access to the AppAnalyticsQueryRequest object, and consider [creating a permission set](#) for them. By default, admins have the permissions required to request package log files and usage summaries using the AppAnalyticsQueryRequest object.

In a 24-hour period, you can download a maximum 20 GB of AppExchange App Analytics data.

Package usage summary data is available to download for 10 years from the summary file log date. Package usage log data is available to download for 45 days from the date the log event occurred. Subscriber snapshot data is available to download for 45 days from the snapshot date.

 **Note:** AppExchange App Analytics is subject to certain usage restrictions as described in the [AppExchange Program Policies](#).

1. Log into the License Management Org (LMO) that the package is registered to.
2. From the LMO, complete the required fields in the [AppAnalyticsQueryRequest](#) in SOAP API.
3. Retrieve the App Analytics Query Request object created in the API request. The `DownloadURL` field populates after the request is completed.
4. Click the URL in the `DownloadURL` field in the App Analytics Query Request object, and download the .csv file.

 **Note:** The download URL expires after 60 minutes.

## AppExchange App Analytics Best Practices

To plan and maximize your AppExchange App Analytics query strategy, follow our best practices. First, use file compression to reduce your data results file size. Second, schedule and automate your regular App Analytics queries. Third, plan, schedule, and automate catch-up queries to supplement your regular query data.

### [How Does AppExchange App Analytics Data Flow?](#)

As your customers use your managed packages, they produce data. Their usage data is collected daily in our data lake from each Salesforce instance. Usage data arrives to our data lake throughout the day. From time to time, there can be occasional data arrival delays. Also, data builds and timestamps vary by data type. For these reasons, to optimize your data retrieval, plan out your AppExchange App Analytics query strategy.

### [How Should I Plan My App Analytics Query Strategy?](#)

Your detailed query strategy depends on the size and scope of your business and the data types that you're querying.

### [Recommendations](#)

Your query strategy varies based on your business size and scope. Also, your query strategy must adapt as your business grows. To stay current, follow our App Analytics query recommendations for small, medium, and large-sized partners.

### [Where Do I Go for More Information About AppExchange App Analytics Queries?](#)

Questions are natural when you start automating your queries and planning your query strategy. To find a good solution when you have questions, review your code base and the size and skill of your development team.

#### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

## How Does AppExchange App Analytics Data Flow?

As your customers use your managed packages, they produce data. Their usage data is collected daily in our data lake from each Salesforce instance. Usage data arrives to our data lake throughout the day. From time to time, there can be occasional data arrival delays. Also, data builds and timestamps vary by data type. For these reasons, to optimize your data retrieval, plan out your AppExchange App Analytics query strategy.

Because Salesforce instances are located around the world, the time of data collection varies by region. EU (EMEA) data arrives first, then North America (NA) data. Data from Asia Pacific (AP) instances arrives last.

Our AppExchange App Analytics jobs run on local instance times on a non-peak schedule. Depending on when you query for your data and where your customers are located, sometimes you retrieve 100% of your data at one time. Other times you must issue more queries to retrieve it all.

#### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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



Data delivery to and arrival in our data lake also depends on factors that can affect a given instance, such as the health of the instance or technical dependencies. Ordinarily you can expect all your org data to arrive in the data lake by 23:00 Coordinated Universal Time (UTC) the day after it was recorded. However, occasionally, there can be delays.

Each AppExchange App Analytics data type is also compiled at different times:

Data Type	Build Information	Example
Subscriber Snapshots	<ul style="list-style-type: none"> <li>Snapshots use data collected at approximately 01:00 instance local time.</li> <li>Snapshots are generated nightly at approximately 03:00 instance local time.</li> <li>All timestamps are normalized to 00:00 UTC of that day.</li> </ul>	<p>For the March 1, 2021 snapshot:</p> <ul style="list-style-type: none"> <li>All records have this timestamp: 2021-03-01T00:00:00Z.</li> <li>All data normally arrives by March 2, 2021 23:00 UTC.</li> </ul>
Package Usage Summaries	<ul style="list-style-type: none"> <li>Summaries use data collected for an entire month.</li> <li>Summaries are built monthly.</li> <li>All timestamps are normalized to 00:00 UTC on the last day of the month.</li> </ul>	<p>For the March 2021 summary available on April 1, 2021:</p> <ul style="list-style-type: none"> <li>All records have this timestamp: 2021-03-31T00:00:00Z.</li> <li>All data normally arrives by April 1, 2021 23:00 UTC.</li> </ul>
Package Log Files	<ul style="list-style-type: none"> <li>Log files use data from the previous day.</li> <li>Log files are generated nightly at approximately 05:00 instance local time.</li> </ul>	<p>For the March 1, 2021 log file:</p> <ul style="list-style-type: none"> <li>All records have precise timestamps associated with when that log event occurred.</li> <li>All data normally arrives by March 2, 2021 23:00 UTC.</li> </ul>

## How Should I Plan My App Analytics Query Strategy?

Your detailed query strategy depends on the size and scope of your business and the data types that you're querying.

All partners can take advantage of these query strategies:

- Choose a data results `FileType` value, and select a corresponding `FileCompression`, which allows you to request `gzip` compression for `csv` files or `snappy` column compression for `parquet` files.
- Create regularly scheduled, automated queries.
- To sweep in late-arriving data, create catch-up queries using the `AvailableSince` field.

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

## Compress Your Results Files

Your App Analytics query plan starts with your results file type and file compression. Data can eat up time and space, so do more with less by specifying the type of file you download. Reduce your data download time by specifying how your results file is compressed.

If you don't specify file type or file compression, your results file defaults to `csv` with no compression for backwards compatibility reasons. If you choose the `parquet` file type, your results file includes data type information for each column.

We recommend that you always compress your results files. Choose from these SOAP API `AppAnalyticsQueryRequest` `FileType` and `FileCompression` value combinations:

FileType	FileCompression
<code>csv</code> (default)	<ul style="list-style-type: none"><li>• <code>none</code> (default when <code>FileType</code> is <code>csv</code>)</li><li>• <code>gzip</code></li></ul>
<code>parquet</code>	<ul style="list-style-type: none"><li>• <code>snappy</code> (default when <code>FileType</code> is <code>parquet</code>)</li><li>• <code>gzip</code></li><li>• <code>none</code></li></ul>



**Note:** When you download your App Analytics query result data, the HTTP response contains one or two important headers. The `Content-Type` header indicates the file type (`txt/csv` or `application/parquet`). For queries with `csv` `FileType` and `gzip` `FileCompression`, the `Content-Encoding` header indicates `gzip` encoding. Modern browsers often decode the `gzip`-encoded file automatically which results in a saved, uncompressed `.csv` file. Regardless if the file is automatically decoded or not, its filename extension is `.csv`.

## Schedule and Automate Your Queries

After you determine what queries to run and how often to run them, you want to schedule those queries. The easiest way is via automation.

What do we mean by automation? Write code that creates query request records on your schedule, monitors them, retrieves the data, and stores your AppExchange App Analytics data somewhere. For example, you can store the data in a custom object in your License Management Org.

Your automation options include, but aren't limited to:

- Custom API integrations using REST or SOAP API calls
- Salesforce DX automation using the CLI

- Salesforce flows
- Apex triggers

For example, automate the retrieval of package usage summaries using Apex triggers.

If you want to also automate the retrieval of package usage log data, look to a different storage solution that scales with the data volume the logs contain.

## Create Catch-Up Queries

A catch-up query is like a broom, sweeping for data newly added to our data lake. Catch-up queries rely on you already having regular queries in place.

For example, on March 2, 2021 18:00 UTC you run this regular query that retrieves package usage log data for March 1, 2021:

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-01T00:00:00Z
EndTime=2021-03-02T00:00:00Z
DataType=PackageUsageLog
FileType=csv
FileCompression=gzip"
```


Rerun that exact same query on March 3, 2021 18:00 UTC, but add the `AvailableSince` field set to the day and time you ran your original query: 2021-03-02T18:00:00Z. This query is your ad hoc catch-up query. It retrieves any data newly added to the data lake for March 2 since you ran your regular query:

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-01T00:00:00Z
EndTime=2021-03-02T00:00:00Z
DataType=PackageUsageLog
FileType=csv
FileCompression=gzip
AvailableSince=2021-03-02T18:00:00Z"
```

You can use catch-up queries in many different ways, which we discuss in more detail in the Recommendations section.

When creating catch-up queries, keep these considerations in mind.

- If `StartTime` is specified, the `AvailableSince` date must be later.
- If `EndTime` is specified, the `AvailableSince` date must be later.
- All queries must include `StartTime` or `AvailableSince` or both.
- `AvailableSince` must be earlier than now.

 **Note:** What happens when you want to create an ad hoc catch-up query, but you forgot when you ran the original query? Use Salesforce CLI and your original query's `sObjectID` to look up the `QuerySubmittedTime`, like this: `sfdx force:data:record:get --subjecttype AppAnalyticsQueryRequest --subjectid 0XIxxxxxxxxxxxxxxxxx` Set your ad hoc catch-up query `AvailableSince` value to equal the `QuerySubmittedTime`.


SEE ALSO:

[Apache Parquet](#)

[Automate AppAnalytics - AWS Stack](#)

Recommendations

Your query strategy varies based on your business size and scope. Also, your query strategy must adapt as your business grows. To stay current, follow our App Analytics query recommendations for small, medium, and large-sized partners.

 **Note:** In the unlikely event of data delays, we regenerate data for log events that happened up to 30 days in the past. To ensure that you consistently retrieve the most complete data, we recommend that you schedule catch-up queries that look back 30 days.

Small-Sized Partners

Small-sized partners have manageable subscriber bases and one or two managed packages. A small partner’s total daily usage data across all managed packages is 5 GB or less. Also, small partner’s queries complete well under the 15-minute processing time limit.

Medium-Sized Partners

Medium-sized partners have bigger subscriber bases and about six managed packages. A medium-sized partner’s total daily usage data across all managed packages is at or just over 20 GB. Also, this partner’s queries approach or hit the 15-minute processing time limit.

Large-Sized Partners

Large partners have large subscriber bases and many managed packages. A large partner’s total daily usage data is more than 20 GB. Sometimes a large partner’s data from just one managed package is larger than the 20-GB daily limit. Also, large partners often must create a smaller time range for each query to complete in under the 15-minute processing time limit.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Small-Sized Partners

Small-sized partners have manageable subscriber bases and one or two managed packages. A small partner’s total daily usage data across all managed packages is 5 GB or less. Also, small partner’s queries complete well under the 15-minute processing time limit.

Given how manageable smaller partners’ data is, after you run your regular queries one time, we recommend that you run a daily catch-up query as your main query. Sweep in all data for all your managed packages looking back 30 days.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Data Type	How to Get Started	How to Schedule Catch-Up Queries
Subscriber Snapshots	An initial query to retrieve data from when App Analytics was enabled for your managed package.	<ul style="list-style-type: none"><li>• One daily catch-up query.</li><li>• Set <code>AvailableSince</code> to the day and time your last regular query ran.</li><li>• Set <code>StartTime</code> to 30 days ago.</li><li>• Omit <code>EndTime</code>.</li><li>• Each day, advance <code>StartTime</code> and <code>AvailableSince</code> by 1 day.</li></ul>

Data Type	How to Get Started	How to Schedule Catch-Up Queries
Package Usage Summaries	An initial query to retrieve data from when App Analytics was enabled for your managed package.	<ul style="list-style-type: none"> <li>One daily catch-up query.</li> <li>Set <code>AvailableSince</code> to the day and time your last regular query ran.</li> <li>Set <code>StartTime</code> to the first of the previous month.</li> <li>Omit <code>EndTime</code>.</li> <li>Each day, advance <code>AvailableSince</code> by 1 day.</li> <li>Each month, advance <code>StartTime</code> to the first of the previous month.</li> </ul>
Package Log Files	An initial query to retrieve data from when App Analytics was enabled for your managed package.	<ul style="list-style-type: none"> <li>One daily catch-up query.</li> <li>Set <code>AvailableSince</code> to the day and time your last regular query ran.</li> <li>Set <code>StartTime</code> to 30 days ago.</li> <li>Omit <code>EndTime</code>.</li> <li>Each day, advance <code>StartTime</code> and <code>AvailableSince</code> by 1 day.</li> </ul>



**Example:** Most of your customers use your package on an NA or EU instance, so you run your queries at 18:00 UTC. You have a couple customers on an AP instance, so you create catch-up queries to ensure that you capture data from around the world.

1. On March 31 at 18:00 UTC, run your regular queries.

Subscriber Snapshot

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "DataType=SubscriberSnapshot
FileType=csv
FileCompression=gzip
StartTime=2020-03-30T00:00:00Z
EndTime=2020-03-31T00:00:00Z"
```

Package Usage Summary

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "DataType=PackageUsageSummary
FileType=csv
FileCompression=gzip
StartTime=2020-02-01T00:00:00Z
EndTime=2020-03-01T00:00:00Z"
```

Package Log File

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
```

```
--values "DataType=PackageUsageLog  
FileType=csv  
FileCompression=gzip  
StartTime=2020-03-30T00:00:00Z  
EndTime=2020-03-31T00:00:00Z"
```

2. On April 1 at 18:00 UTC run these three catch-up queries.

#### Subscriber Snapshot Catch-Up Query

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "DataType=SubscriberSnapshot  
FileType=csv  
FileCompression=gzip  
StartTime=2020-03-02T00:00:00Z  
AvailableSince=2020-03-31T18:00:00Z"
```

#### Package Usage Summary Catch-Up Query

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "DataType=PackageUsageSummary  
FileType=csv  
FileCompression=gzip  
StartTime=2020-03-01T00:00:00Z  
AvailableSince=2020-03-31T18:00:00Z"
```

#### Package Log File Catch-Up Query

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "DataType=PackageUsageLog  
FileType=csv  
FileCompression=gzip  
StartTime=2020-03-02T00:00:00Z  
AvailableSince=2020-03-31T18:00:00Z"
```

3. On April 2 at 18:00 UTC, run the same catch-up queries, but advance the subscriber snapshot and package log file AvailableSince and StartTime date by 1 day each. Advance the package usage summary AvailableSince by 1 day.

#### Subscriber Snapshot Catch-Up Query

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "DataType=SubscriberSnapshot  
FileType=csv  
FileCompression=gzip  
StartTime=2020-03-03T00:00:00Z  
AvailableSince=2020-04-01T18:00:00Z"
```

#### Package Usage Summary Catch-Up Query

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "DataType=PackageUsageSummary"
```

```
FileType=csv
FileCompression=gzip
StartTime=2020-03-01T00:00:00Z
AvailableSince=2020-04-01T18:00:00Z"
```

#### Package Log File Catch-Up Query

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "DataType=PackageUsageLog
FileType=csv
FileCompression=gzip
StartTime=2020-03-03T00:00:00Z
AvailableSince=2020-04-01T18:00:00Z"
```

## Medium-Sized Partners

Medium-sized partners have bigger subscriber bases and about six managed packages. A medium-sized partner's total daily usage data across all managed packages is at or just over 20 GB. Also, this partner's queries approach or hit the 15-minute processing time limit.

We recommend that after you run your regular queries one time, use catch-up queries as your main queries for subscriber snapshots and package usage summaries. Use a combination of daily queries and catch-up queries for package log files.

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Data Type	How to Get Started	How to Schedule Catch-Up Queries
Subscriber Snapshots	An initial query to retrieve data from when App Analytics was enabled for your managed packages.	<ul style="list-style-type: none"> <li>One daily query.</li> <li>Set <code>AvailableSince</code> to the day and time your last regular query ran.</li> <li>Set <code>StartTime</code> to 30 days ago.</li> <li>Omit <code>EndTime</code>.</li> <li>Each day, advance <code>StartTime</code> and <code>AvailableSince</code> by 1 day.</li> </ul>
Package Usage Summaries	An initial query to retrieve data from when App Analytics was enabled for your managed packages.	<ul style="list-style-type: none"> <li>One daily catch-up query.</li> <li>Set <code>AvailableSince</code> to the day and time your last regular query ran.</li> <li>Set <code>StartTime</code> to the first of the previous month.</li> <li>Omit <code>EndTime</code>.</li> </ul>

Data Type	How to Get Started	How to Schedule Catch-Up Queries
		<ul style="list-style-type: none"> <li>Each day, advance <code>AvailableSince</code> by 1 day.</li> <li>Each month, advance <code>StartTime</code> to the first of the previous month.</li> </ul>
Package Log Files	One regular daily query per package.	<ul style="list-style-type: none"> <li>One daily catch-up query per package.</li> <li>Set <code>AvailableSince</code> to the day and time your last regular query ran.</li> <li>Set <code>StartTime</code> to 30 days ago.</li> <li>Set <code>EndTime</code> equal to the <code>StartTime</code> of your regular query.</li> <li>Each day, advance <code>StartTime</code>, <code>EndTime</code>, and <code>AvailableSince</code> by 1 day.</li> </ul>



**Example:** Half of your customers use your package on an NA or EU instance, so you run your regular queries at 18:00 UTC. The other half of your customers are on an AP instance, so you create catch-up queries to ensure that you capture data from around the world.

- On March 31 at 18:00 UTC, run your regular package log file queries for each of your packages.

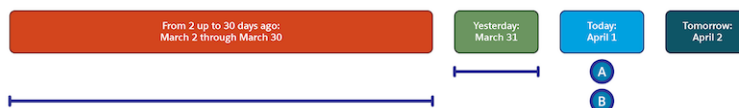
Package 1

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-30T00:00:00Z
EndTime=2021-03-31T00:00:00Z
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX
FileType=csv
FileCompression=gzip"
```

Package 2

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-30T00:00:00Z
EndTime=2021-03-31T00:00:00Z
DataType=PackageUsageLog
PackageIds=0337XXXXXXXXXX
FileType=csv
FileCompression=gzip"
```

- On April 1 at 18:00 UTC onwards, run regular and catch-up package log file queries.



## A. Regular Queries

### Package 1

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-31T00:00:00Z
EndTime=2021-04-01T00:00:00Z
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX
FileType=csv
FileCompression=gzip"
```

### Package 2

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-31T00:00:00Z
EndTime=2021-04-01T00:00:00Z
DataType=PackageUsageLog
PackageIds=0337XXXXXXXXXX
FileType=csv
FileCompression=gzip"
```

## B. Catch-Up Queries

### Package 1

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-01T00:00:00Z
EndTime=2021-03-31T00:00:00Z
AvailableSince=2021-03-31T18:00:00Z
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX
FileType=csv
FileCompression=gzip"
```

### Package 2

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-01T00:00:00Z
EndTime=2021-03-31T00:00:00Z
AvailableSince=2021-03-31T18:00:00Z
DataType=PackageUsageLog
PackageIds=0337XXXXXXXXXX
FileType=csv
FileCompression=gzip"
```

- On April 2, repeat the same queries that you ran on April 1, but advance the queries by a day.



## A. Regular Queries

Package 1

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-04-01T00:00:00Z
EndTime=2021-04-02T00:00:00Z
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX
FileType=csv
FileCompression=gzip"
```

Package 2

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-04-01T00:00:00Z
EndTime=2021-04-02T00:00:00Z
DataType=PackageUsageLog
PackageIds=0337XXXXXXXXXX
FileType=csv
FileCompression=gzip"
```

B. Catch-Up Queries

Package 1

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-02T00:00:00Z
EndTime=2021-04-01T00:00:00Z
AvailableSince=2021-04-01T18:00:00Z
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX
FileType=csv
FileCompression=gzip"
```

Package 2

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2020-03-02T00:00:00Z
EndTime=2021-04-01T00:00:00Z
AvailableSince=2021-04-01T18:00:00Z
DataType=PackageUsageLog
PackageIds=0337XXXXXXXXXX
FileType=csv
FileCompression=gzip"
```

## Large-Sized Partners

Large partners have large subscriber bases and many managed packages. A large partner's total daily usage data is more than 20 GB. Sometimes a large partner's data from just one managed package is larger than the 20-GB daily limit. Also, large partners often must create a smaller time range for each query to complete in under the 15-minute processing time limit.

Large partners frequently create one query per managed package per 12, 6, or 1-hour increments throughout a 24-hour period. How frequently you schedule your queries really depends on your data volume.


We recommend that you use a combination of queries and multiple catch-up queries for all data types

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Data Type	How to Get Started	How to Schedule Catch-Up Queries
Subscriber Snapshots	One daily query per package.	<ul style="list-style-type: none"> <li>One daily query per package.</li> <li>Set <code>AvailableSince</code> to the day and time your last regular query ran.</li> <li>Set <code>StartTime</code> to 30 days ago.</li> <li>Omit <code>EndTime</code>.</li> <li>Each day, advance <code>StartTime</code> and <code>AvailableSince</code> by 1 day.</li> </ul>
Package Usage summaries	One daily query per package.	<ul style="list-style-type: none"> <li>One daily catch-up query per package.</li> <li>Set <code>AvailableSince</code> to the day and time your last regular query ran.</li> <li>Set <code>StartTime</code> to the first of the previous month.</li> <li>Omit <code>EndTime</code>.</li> <li>Each day, advance <code>AvailableSince</code> by 1 day.</li> <li>Each month, advance <code>StartTime</code> to the first of the previous month.</li> </ul>
Package Log Files	<ul style="list-style-type: none"> <li>To retrieve all your data, create multiple segmented daily, automated App Analytics queries spread throughout the day.</li> <li>Break up your requests by managed package and by time increments throughout the day.</li> </ul>	<p>Create two levels of catch-up queries per day.</p> <ul style="list-style-type: none"> <li>Create one catch-up query per package that sweeps data from 2 days ago.</li> <li>Create a second catch-up query that sweeps data from 3 to 30 days ago.</li> <li>Each day, advance <code>StartTime</code>, <code>EndTime</code>, and <code>AvailableSince</code> by 1 day.</li> </ul>

 **Example:** Your customers use your package on all Salesforce instances around the world, and your managed packages produce significant amounts of data. You schedule queries to run at the same time, each covering a 12-hour period, and you create a layered catch-up query plan to capture data from all instances.

In this example, we show two of your dozens of managed packages.

1. On March 31 at 18:00 UTC, run your regular package log file queries.

Package 1

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-30T00:00:00
EndTime=2021-03-30T12:00:00
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX
FileType=parquet
FileCompression=snappy"
```

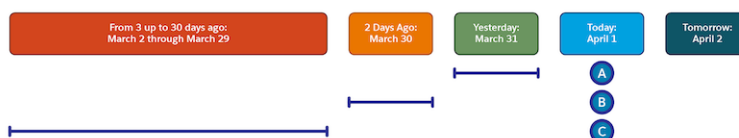
```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-30T12:00:00
EndTime=2021-03-31T00:00:00
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX
FileType=parquet
FileCompression=snappy"
```

Package 2

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-30T00:00:00
EndTime=2021-03-30T12:00:00
DataType=PackageUsageLog
PackageIds=0337XXXXXXXXXX
FileType=parquet
FileCompression=snappy"
```

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-30T12:00:00
EndTime=2021-03-31T00:00:00
DataType=PackageUsageLog
PackageIds=0337XXXXXXXXXX
FileType=parquet
FileCompression=snappy"
```

2. On April 1 at 18:00 UTC, run your regular and catch-up package log file queries.



A. Package Log File Regular Queries

Package 1

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-31T00:00:00Z
EndTime=2021-03-31T12:00:00Z
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX
FileType=parquet
FileCompression=snappy"
```

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-31T12:00:00Z
EndTime=2021-04-01T00:00:00Z
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX
FileType=parquet
FileCompression=snappy"
```

Package 2

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-31T00:00:00Z
EndTime=2021-03-31T12:00:00Z
DataType=PackageUsageLog
PackageIds=0337XXXXXXXXXX
FileType=parquet
FileCompression=snappy"
```

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-31T12:00:00Z
EndTime=2021-04-01T00:00:00Z
DataType=PackageUsageLog
PackageIds=0337XXXXXXXXXX
FileType=parquet
FileCompression=snappy"
```

B. Package Log File 2 Days Ago Catch-Up Queries

Package 1

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-30T00:00:00Z
EndTime=2021-03-31T00:00:00Z
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX
FileType=parquet
FileCompression=snappy
AvailableSince=2020-03-31T18:00:00Z"
```

#### Package 2

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-30T00:00:00Z
EndTime=2021-03-31T00:00:00Z
DataType=PackageUsageLog
PackageIds=0337XXXXXXXXXX
FileType=parquet
FileCompression=snappy
AvailableSince=2020-03-31T18:00:00Z"
```

#### C. Package Log File From 3 to 30 Days Ago Catch-Up Queries

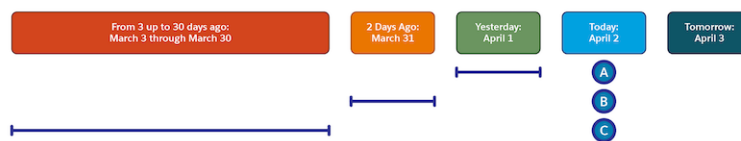
##### Package 1

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-01T00:00:00Z
EndTime=2021-03-30T00:00:00Z
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX
FileType=parquet
FileCompression=snappy
AvailableSince=2020-03-31T18:00:00Z"
```

##### Package 2

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-03-01T00:00:00Z
EndTime=2021-03-30T00:00:00Z
DataType=PackageUsageLog
PackageIds=0337XXXXXXXXXX
FileType=parquet
FileCompression=snappy
AvailableSince=2020-03-31T18:00:00Z"
```

- On April 2 onwards, run your regular and your catch-up package log file queries, advancing the dates by 1 day.



#### A. Package Log File Regular Queries

##### Package 1

```
sfdx force:data:record:create
--subjecttype AppAnalyticsQueryRequest
--values "StartTime=2021-04-01T00:00:00Z
EndTime=2021-04-01T12:00:00Z
DataType=PackageUsageLog
PackageIds=0336XXXXXXXXXX"
```

```
FileType=parquet  
FileCompression=snappy"
```

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "StartTime=2021-04-01T12:00:00Z  
EndTime=2021-04-02T00:00:00Z  
DataType=PackageUsageLog  
PackageIds=0336XXXXXXXXXX  
FileType=parquet  
FileCompression=snappy"
```

#### Package 2

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "StartTime=2021-04-01T00:00:00Z  
EndTime=2021-04-01T12:00:00Z  
DataType=PackageUsageLog  
PackageIds=0337XXXXXXXXXX  
FileType=parquet  
FileCompression=snappy"
```

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "StartTime=2021-04-01T12:00:00Z  
EndTime=2021-04-02T00:00:00Z  
DataType=PackageUsageLog  
PackageIds=0337XXXXXXXXXX  
FileType=parquet  
FileCompression=snappy"
```

#### B. Package Log File 2 Days Ago Catch-Up Queries

##### Package 1

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "StartTime=2021-03-31T00:00:00Z  
EndTime=2021-04-01T00:00:00Z  
DataType=PackageUsageLog  
PackageIds=0336XXXXXXXXXX  
FileType=parquet  
FileCompression=snappy  
AvailableSince=2020-04-01T18:00:00Z"
```

##### Package 2

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "StartTime=2021-03-31T00:00:00Z  
EndTime=2021-04-01T00:00:00Z  
DataType=PackageUsageLog  
PackageIds=0337XXXXXXXXXX  
FileType=parquet
```

```
FileCompression=snappy  
AvailableSince=2020-04-01T18:00:00Z"
```

### C. Package Log File From 3 to 30 Days Ago Catch-Up Queries

#### Package 1

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "StartTime=2021-03-02T00:00:00Z  
EndTime=2021-03-31T00:00:00Z  
DataType=PackageUsageLog  
PackageIds=0336XXXXXXXXXX  
FileType=parquet  
FileCompression=snappy  
AvailableSince=2020-04-01T18:00:00Z"
```

#### Package 2

```
sfdx force:data:record:create  
--subjecttype AppAnalyticsQueryRequest  
--values "StartTime=2021-03-02T00:00:00Z  
EndTime=2021-03-31T00:00:00Z  
DataType=PackageUsageLog  
PackageIds=0337XXXXXXXXXX  
FileType=parquet  
FileCompression=snappy  
AvailableSince=2020-04-01T18:00:00Z"
```

## Where Do I Go for More Information About AppExchange App Analytics Queries?

Questions are natural when you start automating your queries and planning your query strategy. To find a good solution when you have questions, review your code base and the size and skill of your development team.

If you still need help, try these resources:

- If you have an assigned AppExchange Partner Account Manager (PAM) or AppExchange Technical Evangelist (TE), reach out to them.
- Otherwise, go to the Partner Community and post a question to the [ISV TE Experts - Partner Intelligence](#) Chatter group.

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

## Package Usage Summaries

Package usage summaries provide high-level metrics by calendar month. Discover how many users access your package and which operations they perform.

 **Note:** AppExchange App Analytics is subject to certain usage restrictions as described in the [AppExchange Program Policies](#).

AppExchange App Analytics tracks UI, API-based, Lightning-based, and Apex operations and logs each CRUD operation on components and custom objects in packages. Events from sandbox, scratch, and trial orgs aren't tracked in package usage summaries.

Partners and subscribers can access package usage data. Usage summaries become available at the beginning of the subsequent month. For example, you can get the usage summary for May at the beginning of June.


- AppExchange Partners can request monthly usage summaries using the `AppAnalyticsQueryRequest` in SOAP API from the license management org that owns the package.
- Subscribers can download usage summaries from Setup for any package that they installed that passed security review.

### Package Usage Summary

Use the package usage summary to discover how many users access your package and which operation they perform.

## Package Usage Summary

Use the package usage summary to discover how many users access your package and which operation they perform.

 **Note:** In Summer '20, we changed the names of the `DataType` enums in the [AppAnalyticsQueryRequest](#).

Field	Description
<code>custom_entity</code>	The developer name of the component or custom object.
<code>custom_entity_type</code>	The type of component or custom object that the user viewed or manipulated. Examples: <ul style="list-style-type: none"> <li>• <code>ApexClass</code></li> <li>• <code>ApexTrigger</code></li> <li>• <code>CustomObject</code></li> <li>• <code>LightningPage</code></li> <li>• <code>LightningComponent</code></li> <li>• <code>VisualforcePage</code></li> </ul>
<code>managed_package_namespace</code>	Namespace of the package.
<code>month</code>	The month that this usage summary applies to in YYYY-MM format. Example: 2019-03.
<code>num_creates</code>	The number of new records created from the package.
<code>num_deletes</code>	The number of deleted records associated with the package.
<code>num_events</code>	The number of log records associated with a <code>custom_entity_type</code> .
<code>num_reads</code>	The number of records associated with the package that were read.
<code>num_updates</code>	The number of records associated with the package that were updated.
<code>num_views</code>	The count of times the component or page has been viewed.
<code>organization_id</code>	The 15-character ID of the subscriber org.
<code>package_id</code>	The ID of the package.

Field	Description
user_id_token	<p>The hashed token representing the ID of the user who accessed the package. The token persists over time, even if a user's details change. The token also persists across any packages that the user interacts with.</p> <p>The user ID token starts with the prefix 005-. In compliance with privacy regulations, our systems can't access the actual user ID. Likewise, the hashed token can't be linked to the user ID.</p>
user_type	<p>The user license category of the user accessing Salesforce services through the UI or API.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• Guest</li> <li>• Partner</li> <li>• Standard</li> </ul>

## Package Log Files

Analyze adoption and user behavior, then make informed feature development decisions based on data from package log files. AppExchange App Analytics tracks UI, API-based, Lightning-based, and Apex operations, and it logs each CRUD operation on components and custom objects in packages. Events from sandbox and trial orgs are tracked in package log files. Events from scratch orgs aren't tracked.

 **Note:** AppExchange App Analytics is subject to certain usage restrictions as described in the [AppExchange Program Policies](#).

### Package Usage Logs

Make informed development decisions based on package usage log file data. Analyze adoption, user behavior, company information, and Lightning app and page usage data. Usage logs list activity during a 24-hour period, between 12:00 AM and 11:59 PM UTC time.

## Package Usage Logs

Make informed development decisions based on package usage log file data. Analyze adoption, user behavior, company information, and Lightning app and page usage data. Usage logs list activity during a 24-hour period, between 12:00 AM and 11:59 PM UTC time.

 **Note:** In Summer '20 Salesforce changed the names of the Data Type enums in the [AppAnalyticsQueryRequest](#).

Field	Description
api_type	<p>The type of API request.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• E: SOAP Enterprise</li> <li>• O: Old SOAP</li> <li>• P: SOAP Partner</li> <li>• X: XmlRPC</li> <li>• REST: REST</li> </ul>

Field	Description
api_version	The version of the API that's used. Example: 45.0.
app_name	The name of the Lightning application the user accessed. Examples: <ul style="list-style-type: none"> <li>• one:one</li> <li>• FieldServiceApp</li> <li>• Chatter</li> </ul>
class_name	The name of the Apex class. Examples: <ul style="list-style-type: none"> <li>• Help_HomeController</li> <li>• ROAppController_v2</li> <li>• FSL</li> </ul>
cloned_from_organization_id	The ID of the org from which this subscriber org was cloned. Applies to sandbox orgs only. Example: 00Dxx0000000000
custom_entity	The developer name of the component or custom object.
custom_entity_type	The type of component or custom object that the user viewed or manipulated. Examples: <ul style="list-style-type: none"> <li>• ApexClass</li> <li>• ApexTrigger</li> <li>• CustomObject</li> <li>• LightningComponent</li> <li>• LightningPage</li> <li>• VisualforcePage</li> </ul>
entry_point	The entry point of the executed Apex event. <ul style="list-style-type: none"> <li>• GeneralCloner.cloneAndInsertRecords</li> <li>• VF- /apex/CloneUser</li> </ul>
event	The name or ID of the platform event. Examples: <ul style="list-style-type: none"> <li>• /event/011xx0000005akx</li> <li>• SomeCustomEvent</li> </ul>
event_count	The number of platform events consumed by the subscriber. Example: 2.
event_subscriber	The ID of the platform event subscriber. Example: 01qxx0000004Coy.
http_method	The type of HTTP request method. Example: GET.
http_status_code	The HTTP response status code. Example: 404.

Field	Description
login_key	The hashed string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the session expiring. All log lines with the same login key occurred during the same user login session.
log_record_type	Type of log record. Examples: <ul style="list-style-type: none"> <li>• API</li> <li>• ApexExecution</li> <li>• ApexUnexpectedException</li> <li>• ApexSoap</li> <li>• CronJob</li> <li>• PlatformEventConsumer</li> <li>• QueuedExec</li> <li>• RestAPI</li> <li>• URI</li> <li>• VFRemoting</li> <li>• VisualforceRequest</li> </ul>
managed_package_namespace	Namespace of the package.
method_name	The name of the Apex method. Examples: <ul style="list-style-type: none"> <li>• getUserAccessLevelBean</li> <li>• getCurrentDocumentsRates</li> <li>• getAdditionalHelpTemplate</li> </ul>
num_fields	The number of fields accessed by the user in this transaction.
num_soql_queries	The number of SOQL queries completed during the executed Apex event.
operation_count	The number of records accessed by the user in this transaction.
operation_type	The operation performed on the component or custom object. Examples: <ul style="list-style-type: none"> <li>• INSERT</li> <li>• READ</li> <li>• UPDATE</li> <li>• DELETE</li> </ul>
organization_country_code	The ISO-3166 two-character country code corresponding to the subscriber org's address at the time of sign-up. Examples: <ul style="list-style-type: none"> <li>• US</li> </ul>

Field	Description
	<ul style="list-style-type: none"> <li>CA</li> <li>FR</li> </ul>
organization_edition	<p>The name of the Salesforce edition the subscriber org is using.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>Developer Edition</li> <li>Enterprise Edition</li> <li>Unlimited Edition</li> </ul>
organization_id	The 15-character ID of the subscriber org.
organization_instance	<p>The name of the subscriber org's instance.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>AP2</li> <li>EU7</li> <li>NA44</li> </ul>
organization_language_locale	<p>The 2–5 character code that represents the language and locale ISO-639 code of the subscriber org. This code controls the language for the labels displayed in an application.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>de-DE</li> <li>en-US</li> <li>fr-CA</li> </ul>
organization_name	The name of the subscriber org. Example: Acme, Inc.
organization_status	<p>The paid status of the subscriber org.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>Active</li> <li>Demo</li> <li>Free</li> <li>Trial</li> </ul>
organization_time_zone	<p>The default time zone of the subscriber org.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>America/New York</li> <li>America/Los Angeles</li> <li>Europe/Paris</li> </ul>
organization_type	<p>The subscriber org environment type.</p> <p>Examples:</p>

Field	Description
	<ul style="list-style-type: none"> <li>• Production</li> <li>• Sandbox</li> </ul>
package_id	The ID of the package.
package_version_id	The ID of the package version.
page_app_name	<p>The internal name of the Lightning application that the user accessed from the App Launcher.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• LightningSales</li> <li>• Chatter</li> </ul>
page_context	The context of the Lightning page where the event occurred. Example: <code>clients:cardContainer</code> .
page_entity_type	<p>The Lightning entity type of the event.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• Contact</li> <li>• Task</li> </ul>
page_url	The relative URL of the top-level Lightning Experience or Salesforce mobile app page that the user accessed. The page can contain one or more Lightning components. Multiple record IDs can be associated with page_url. Example: <code>/sObject/0064100000JXITSASS/view</code>
parent_ui_element	The parent scope of the Lightning page element where the event occurred. Example: <code>ChatterFeed</code> .
prevpage_url	The relative URL of the previous Lightning Experience or Salesforce mobile app page that the user opened. Example: <code>/sObject/0064100000</code>
quiddity	<p>The type of outer execution associated with the executed Apex event.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• A: QueryLocator Batch Apex</li> <li>• B: Bulk API and Bulk API 2.0</li> <li>• BA: Batch Apex</li> <li>• C: Scheduled Apex</li> <li>• E: Inbound Email Service</li> <li>• F: Future</li> <li>• H: Apex REST</li> <li>• I: Invocable Action</li> <li>• K: Quick Action</li> <li>• L: Lightning</li> <li>• M: Remote Action</li> </ul>

Field	Description
	<ul style="list-style-type: none"> <li>Q: Queueable</li> <li>R: Synchronous Uncategorized</li> <li>S: Serial Batch Apex</li> <li>TA: Tests Async</li> <li>TD: Tests Deployment</li> <li>TS: Tests Synchronous</li> <li>V: Visualforce</li> <li>W: SOAP Webservices</li> <li>X: Execute Anonymous</li> </ul>
referrer_uri	<p>The referring URI from the HTTP request. URIs are redacted in the following ways:</p> <ul style="list-style-type: none"> <li>Query strings are removed.</li> <li>User IDs display as hashed tokens.</li> <li>Subscriber-created URIs, such as VisualForce pages, are removed.</li> </ul>
related_list	<p>A section of a record or other detail page that lists items related to that record.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>Open Activities</li> <li>Stage History</li> </ul>
request_id	<p>The ID of the HTTP request made to the server by the browser. If multiple log lines have the same request ID, they all occurred as part of the same user interaction.</p>
request_size	<p>The size of the callout request body in bytes.</p>
request_status	<p>The status of the HTTP request for the page or action that accesses a component or custom object in a package.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>A = Auth Error</li> <li>F = Failure</li> <li>N = 404 error</li> <li>R = Redirect</li> <li>S = Success</li> <li>U = Undefined</li> </ul>
response_size	<p>The size of the callout response in bytes.</p>
rows_processed	<p>The number of rows that were processed in the request.</p>
session_key	<p>The HTTP session ID for the HTTP request to access a component or custom object in a package. The session ID is hashed.</p>
stack_trace	<p>The stack trace associated with the Apex exception.</p>

Field	Description
target_ui_element	<p>The Lightning target page element where the event occurred.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• label body truncate</li> <li>• tabitem-link</li> </ul>
timestamp_derived	<p>The access time of a component or custom object in a package in ISO8601-compatible format (YYYY-MM-DDTHH:MM:SS.sssZ). Example: 2018-07-27T11:32:59.555Z.</p>
ui_event_sequence_num	<p>An auto-incremented sequence number of the current Lightning event since the session started.</p>
ui_event_source	<p>The user action on the Lightning record or records. This value indicates whether the user's action was on a single record or multiple records. For example, <code>read</code> indicates that one record was read, such as on a record detail page. In contrast, <code>reads</code> indicates that multiple records were read, such as in a list view.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• click</li> <li>• create</li> <li>• delete</li> <li>• hover</li> <li>• read</li> <li>• update</li> </ul>
ui_event_type	<p>The type of Lightning event.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• crud</li> <li>• system</li> <li>• user</li> </ul>
url	<p>The redacted URL of the request to access a component or custom object in a package. URLs are redacted in the following ways.</p> <ul style="list-style-type: none"> <li>• Query strings are removed.</li> <li>• User IDs display as hashed tokens.</li> <li>• Subscriber-created URLs, such as VisualForce pages, are removed.</li> </ul> <p>For Lightning-based URLs, only <code>/aura</code> is displayed. For Visualforce-based URLs that aren't pages owned by the managed package, either <code>/apex</code> or <code>/apexrest</code> is displayed.</p>
user_agent	<p>The browser and operating system of the device used to make the request.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• Mozilla/5.0 (iPhone; CPU iPhone OS 12_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) CriOS/69.0.3497.105 Mobile/15E148 Safari/605.1</li> <li>• Mozilla/5.0 (Linux; Android 8.0.0; SM-G960F Build/R16NW) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.84 Mobile Safari/537.36</li> </ul>

Field	Description
<code>user_country_code</code>	The default ISO-3166 two-character country code of the user.  Examples: <ul style="list-style-type: none"><li>• CA</li><li>• FR</li><li>• US</li></ul>
<code>user_id_token</code>	The hashed token representing the ID of the user who accessed the package. The ID persists, even if a user's details change. The token also persists across any packages that the user interacts with.  The user ID token starts with the prefix 005-. In compliance with privacy regulations, our systems can't access the actual user ID. Likewise, the hashed token can't be linked to the user ID.
<code>user_time_zone</code>	The default time zone of the user. Example: <code>America/New_York</code> .
<code>user_type</code>	The user license category of the user accessing Salesforce services through the UI or API.  Examples: <ul style="list-style-type: none"><li>• Guest</li><li>• Partner</li><li>• Standard</li></ul>

## Subscriber Snapshots

Subscriber snapshots provide a point-in-time summary of your subscribers' activity. Use subscriber snapshots to see usage trends by org and package over time.

 **Note:** AppExchange App Analytics is subject to certain usage restrictions as described in the [AppExchange Program Policies](#).

AppExchange App Analytics takes a daily snapshot of org, package, and custom entity data. Snapshots are captured daily at 00:00 UTC and become available for download immediately thereafter. You request a date and time, or range of dates and times, and you receive one snapshot per valid date and time requested. For example, if on April 7, 2020 you request a date and time range of `StartTime=2020-04-04T00:00:00Z` `EndTime=2020-04-06T23:59:59Z`, you receive three snapshots, one for each completed day.

 **Note:** Starting in Summer '20 we changed the names of the `DataType` enums in the [AppAnalyticsQueryRequest](#).

Field	Description
<code>custom_entity</code>	The developer name of the component or custom object.  Examples: <ul style="list-style-type: none"><li>• Amount</li><li>• Travel_Expense</li></ul>

Field	Description
date	The subscriber snapshot date requested, in YYYY-MM-DDT00:00:00Z format. Each point-in-time snapshot is captured at 00:00 UTC on the date specified. Example: 2020-04-04T00:00:00Z
managed_package_namespace	Namespace of the package. Example: sfdx_isv_pkg001
organization_edition	The name of the Salesforce edition the subscriber org is using. Examples: <ul style="list-style-type: none"><li>• Developer Edition</li><li>• Enterprise Edition</li><li>• Unlimited Edition</li></ul>
organization_id	The 15-character ID of the subscriber org. Example: 00D4m000000Td8Y.
organization_name	The name of the subscriber org. Example: My_Org.
organization_status	The paid status of the subscriber org. Examples: <ul style="list-style-type: none"><li>• ACTIVE</li><li>• DEMO</li><li>• FREE</li><li>• TRIAL</li></ul>
package_id	The ID of the managed package. Example: 033xx00000000CI.
package_version_id	The ID of the managed package version. Example: 04t6A0000004eytQAA.
record_count	Total records for the custom entity in that org on the specified snapshot date.

## Test Custom Integrations

To test your custom integrations in a nonproduction environment, use AppExchange App Analytics Simulation Mode. Submit an App Analytics query request and receive sample usage data.



**Note:** AppExchange App Analytics is subject to certain usage restrictions as described in the [AppExchange Program Policies](#).

To receive sample usage data, enable simulation mode, then submit a query request that includes a simulation mode package ID.


1. Enable simulation mode in your test org using the Metadata API [AppAnalyticsSettings](#) `enableSimulationMode` org preference.
2. To simulate package usage log, usage summary, or subscriber snapshot downloads, complete the required fields in your SOAP API [AppAnalyticsQueryRequest](#). Include `DataType`, and leave `OrganizationIDs` blank. For `PackageIDs`, include at least one simulation mode package ID that matches the scenario you're testing.

### USER PERMISSIONS

To enable simulation mode:

- `ModifyMetadata`

Package Type	Simulation Mode Package ID	Description
Small Data Set	033xx00SIMsmall	Contains a small amount of data. For use with all query types. Use this package ID to download data for any query-allowed timespan.
Large Data Set	033xx00SIMlarge	Contains a large amount of data for two org IDs (00Dxx00SIM00foo and 00Dxx00SIM00bar). For use only with package usage log queries.
Empty Data Set	<p>Use any other 15-character package ID prefixed with 033xx00SIM.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>033xx00SIMempty</li> <li>033xx00SIM44444</li> </ul>	Contains no data. For use with all query types. Use one of these package IDs to return an empty data set.

- Submit your query.
  - Check your API request, then do one of the following:
    - Upon success, retrieve the App Analytics Query Request object created in the API request. The `DownloadURL` field populates when the request is completed.
    - Upon error, edit your query. Use a smaller time window, such as a 14 days, or specify one org ID. Then resubmit your query.
  - Download the .csv file containing sample usage data from the `DownloadURL` field in the App Analytics Query Request object.
-  **Important:** When simulation mode is enabled, you can only access our sample usage data. Disable simulation mode to access your production data.

## CHAPTER 10 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.

 **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.

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

### 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	<p>Contains deal-specific information about an active product on an order, along with the corresponding customer details.</p> <p>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.</p>
Partner Pricebook Entry	<p>Contains one or more products from a catalog.</p> <p>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.</p>
Service Order Log	<p>Stores information about the performance of the COA for debugging purposes.</p> <p>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.</p>

## 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.	<p>The customer's contract renewal date.</p> <p>Notify Salesforce of the reduction according to the terms of your partnership agreement,</p>

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: <ul style="list-style-type: none"> <li>• ISVforce orders that don't provision licenses in a customer's org</li> <li>• Processed OEM orders that have a future start date</li> <li>• All OEM and ISVforce cancellation and reduction orders</li> </ul>
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.

Order Status	Possible Actions				
	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: <ul style="list-style-type: none"> <li>Setting up the COA</li> <li>Assigning access to the COA</li> <li>Building custom integrations using COA objects</li> <li>Serving as the context user for the COA email service</li> </ul>

## 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.

- 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**.

### USER PERMISSIONS

- To assign a permission set:
- Assign Permissions Sets

## 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.

- 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.

### USER PERMISSIONS

- To configure Apex email services and email service addresses:
- Modify All Data

Field	Value
Email Service Name	<i>SFDC Service Order</i>
Apex Class	<i>ProcessServiceOrderStatus</i>
Accept Attachments	<b>Text attachments only</b>
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	<i>SFDC_Service_Order</i>
Active	Select to enable
Context User	Select a Salesforce admin in your org

- For **Accept Email From**, remove the autopopulated email address. This field must be blank. Otherwise, the email service can't connect to Salesforce.
- Click **Save**. Salesforce generates a unique address for the email service (1), which the COA uses to sync your product data.

SETUP  
Email Services

Edit Activate Deactivate Delete Cancel

Email Service Name: **SFDC Service Order**

Apex Class: **ProcessServiceOrderStatus**

Accept Attachments: **Text attachments only**

Advanced Email Security Settings: ☐ ⓘ

Accept Email From: **All email addresses (subject to security settings)**

Convert Text Attachments to Binary Attachments: ☐ ⓘ

Active: ☒

▼ Failure Response Settings

Over Email Rate Limit Action: Discard message

Deactivated Email Address Action: Discard message

Deactivated Email Service Action: Discard message

Unauthenticated Sender Action: Discard message

Unauthorized Sender Action: Discard message

Enable Error Routing: ☐ ⓘ

Email Addresses New Email Address

Action	Email Address	Context User
<a href="#">View</a>   <a href="#">Edit</a>	sfdc service order@...	

- 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](#).

- Log in to the org where the COA is installed.
- Open the App Launcher.

### USER PERMISSIONS

To manage custom apps:

- Customize Application

To import product data:

- COA Admin User

- 3. Under All Items, click **COA Setup**.
- 4. Go to Production Settings, and provide your username, API key, and activation code.

Production Settings

Test Settings

Set up a production connection using the credentials you received from Partner Operations. After you connect, your contract and product catalog data is imported to the Channel Order App and you can send orders to Salesforce for billing and activation.

\* Username:

mrigsby@

\* API Key :

\* Activation Code :

.....

Save

- 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.

- 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**.

### USER PERMISSIONS

To create and edit custom tabs:

- Customize Application

## 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.

- 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**.

### USER PERMISSIONS

To create and edit custom objects:

- Customize Application

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.

### [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.

### 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.

### 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.

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.

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

## 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.

1. Log in to the org where the COA is installed.

### USER PERMISSIONS

To assign a permission set:

- Assign Permissions Sets

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.


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	<b>Customer</b>
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.

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**.

### USER PERMISSIONS

To create and edit custom tabs:

- Customize Application

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

### USER PERMISSIONS

To manage custom apps:

- Customize Application

## 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.

### 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.

### 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.

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**.

### USER PERMISSIONS

To submit orders:

- COA User
- OR
- COA Admin User

New Initial Order

**1** Customer Information

Company

\* Company Name Ursa Major Solar, Inc.

\* Org ID

Related Opportunity Search opportunities...

Billing Address

\* Country United States

Street One Market, 1st St #300

City San Francisco

\* State California \* Zip/Postal Code 94105

**2** Contract Type

OEM contract

**3** Order Terms & Conditions

☒ Standard terms ☐ Custom terms

Contract Length 12 (months) Renewal Terms 12 (months) Cancellation Terms 30 (days)

Billing Frequency 1 (months) Contract Auto Yes Renew

Cancel ☒ Save Draft Next

6. Select products for the order, and click **Next**.

New Initial Order

API

PRODUCT	APP	PRICING	UNIT	PRICEBOOK
Additional API Calls - 10K per day		FIXED	Org	Additional API Calls - 1,000 per day - OEM <input checked="" type="checkbox"/>

Back ☒ Save Draft Next

7. Adjust the license quantities and, optionally, pricing, and then click **Next**.

**New Initial Order**

▼ Additional API Calls - 1,000 per day - OEM

PRODUCT	UNIT	PRICING	LICENSES TO ADD	PRICE (UNIT/MONTH)	TOTAL PRICE
Additional API Calls - 10K per day	Org	FIXED	10	n/a	n/a

Previous Total: 0      Total to Add: 10      New Total: 10

Back    Save Draft    Next

8. Enter the service and order dates (1), and then review and accept the terms and conditions (2).

**New Initial Order**

**Service and Order Dates**

1. Service Start Date: 02/05/2018

\* Date Partner Received Customer Order: 01/01/2018

\* Date Customer Accepted Salesforce Service Agreement: 01/02/2018

**Terms & Conditions**

General Terms & Conditions

By submitting this Service Order, you warrant that you are at least 18 years old and that you have the legal capacity and

☒ I agree to the general terms & conditions (View Larger)

Product-Specific Terms & Conditions

This product cannot be provisioned without at least one Force.com User subscription purchased from Reseller per

☒ I agree to the specific terms & conditions for this product: Additional API Calls - 10K per day (View Larger)

2

**Order Summary**

▼ Additional API Calls - 1,000 per day - OEM

Back    Save Draft    Submit

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.

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](#).

### USER PERMISSIONS

To clone orders:

- COA User
- OR
- COA Admin User

## 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
<b>Label</b> Created with New COA	<b>Type</b> boolean
<b>Name</b> Created_with_new_COA__c	<b>Properties</b> Create, Defaulted on create, Filter, Group, Sort, Update
	<b>Description</b> Indicates that you’re using the latest version of the Channel Order App (COA). To ensure that your order is processed, check this field.
<b>Label</b> Contract	<b>Type</b> reference
<b>Name</b> Partner_Contract_Rules__c	<b>Properties</b> Create, Filter, Group, Nillable, Sort, Update
	<b>Description</b> Lookup to the related contract terms record. This field is required.
<b>Label</b> Customer Name	<b>Type</b> reference
<b>Name</b> Customer__c	<b>Properties</b> Create, Filter, Group, Nillable, Sort, Update

Field	Details
	<b>Description</b> Lookup to a customer record. Specify an existing customer record. You can't populate customer details using the API. This field is required.
<b>Label</b> Date Partner Received Customer Order	<b>Type</b> date
<b>Name</b> Date_Partner_Received_Customer_Order__c	<b>Properties</b> Create, Filter, Group, Nillable, Sort, Update
	<b>Description</b> Date you received the order from the customer. This field is required.
<b>Label</b> Date Customer Accepted SFDC Service Agreement	<b>Type</b> date
<b>Name</b> Date_Customer_Accepted_SFDC_Svc_Agmt__c	<b>Properties</b> Create, Filter, Group, Nillable, Sort, Update
	<b>Description</b> Date the customer accepted the Salesforce service agreement. This field is required for OEM contracts.
<b>Label</b> Error Comment	<b>Type</b> textarea
<b>Name</b> Error_Comment__c	<b>Properties</b> Create, Nillable, Sort, Update
	<b>Description</b> Stores comments or instructions from Salesforce Partner Operations when a submitted order can't be processed.
<b>Label</b> I Certify a Corresponding Order is Rec'd	<b>Type</b> picklist
<b>Name</b> I_certify__c	<b>Properties</b> Create, Filter, Group, Nillable, Sort, Update
	<b>Description</b> Confirmation that the order was received. Possible values are Yes and No. This field is required.
<b>Label</b> Order Type	<b>Type</b> picklist
<b>Name</b> Order_Type__c	<b>Properties</b> Create, Filter, Group, Nillable, Sort, Update
	<b>Description</b> 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.
<b>Label</b> Service Order Status	<b>Type</b> picklist
<b>Name</b> Service_Order_Status__c	<b>Properties</b> Create, Defaulted on create, Filter, Group, Nillable, Sort, Update
	<b>Description</b> 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.
<b>Label</b> Service Start Date	<b>Type</b> date
<b>Name</b> Service_Start_Date__c	<b>Properties</b> Create, Filter, Group, Sort, Update
	<b>Description</b> 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
<b>Label</b> App	<b>Type</b> string
<b>Name</b> Application__c	<b>Properties</b> Create, Filter, Group, Nillable, Sort
	<b>Description</b> Name of the app associated with the product.
<b>Label</b> Billing Frequency	<b>Type</b> double


Field Name	Details
<b>Name</b> pc_Billing_Frequency__c	<b>Properties</b> Create, Filter, Nillable, Sort, Update  <b>Description</b> How often the customer is billed per year. This value must match your Salesforce contract, unless you've been granted override permissions.
<b>Label</b> Cancellation Terms (days)	<b>Type</b> double
<b>Name</b> pc_Cancellation_Terms__c	<b>Properties</b> Create, Filter, Nillable, Sort, Update  <b>Description</b> Number of days the customer has to cancel the contract. This value must match your Salesforce contract, unless you've been granted override permissions.
<b>Label</b> Contract Auto Renew	<b>Type</b> picklist
<b>Name</b> pc_Contract_Auto_Renew__c	<b>Properties</b> Create, Filter, Group, Nillable, Sort, Update  <b>Description</b> 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.
<b>Label</b> Contract Length	<b>Type</b> double
<b>Name</b> pc_Contract_Length__c	<b>Properties</b> Create, Filter, Nillable, Sort, Update  <b>Description</b> Length of the contract in months. This value must match your Salesforce contract, unless you've been granted override permissions.
<b>Label</b> Currency	<b>Type</b> string
<b>Name</b> Currency__c	<b>Properties</b> Filter, Nillable, Sort  <b>Description</b> The default contract currency from the contract terms associated with this order. Read-only.
<b>Label</b> Customer Price	<b>Type</b> double
<b>Name</b> Customer_Price_Per_Month__c	<b>Properties</b> Create, Filter, Nillable, Sort, Update

Field Name	Details
	<b>Description</b> Price per unit per month. This field is required for PNR products.
<b>Label</b> Fixed Price <b>Name</b> pc_Fixed_Price__c	<b>Type</b> double <b>Properties</b> Create, Filter, Nillable, Sort, Update <b>Description</b> Fixed price of the product at the time the order was created. This field must be explicitly set when using the API.
<b>Label</b> Partner Contract Term <b>Name</b> pc_Partner_Contract_Term__c	<b>Type</b> reference <b>Properties</b> Create, Filter, Group, Nillable, Sort, Update <b>Description</b> Lookup to the related contract terms record.
<b>Label</b> PNR % <b>Name</b> pc_PNR__c	<b>Type</b> double <b>Properties</b> Create, Filter, Nillable, Sort, Update <b>Description</b> Percent net revenue of the product at the time the order was created. This field must be explicitly set when using the API.
<b>Label</b> Pricing <b>Name</b> pc_Pricing_Type__c	<b>Type</b> picklist <b>Properties</b> Create, Filter, Group, Nillable, Sort, Update <b>Description</b> Pricing model of the product. Possible values are <code>Fixed</code> and <code>PNR</code> . This field must be explicitly set when using the API.
<b>Label</b> Product <b>Name</b> Product_Name__c	<b>Type</b> reference <b>Properties</b> Create, Filter, Group, Nillable, Sort, Update <b>Description</b> Lookup to the related product catalog record.

Field Name	Details
<b>Label</b> Product ID	<b>Type</b> string
<b>Name</b> pc_Product_ID__c	<b>Properties</b> Create, Filter, Group, Nillable, Sort, Update
	<b>Description</b> ID of the product. This field must be explicitly set when using the API.
<b>Label</b> Renewal Terms (months)	<b>Type</b> double
<b>Name</b> pc_Renewal_Terms__c	<b>Properties</b> Create, Filter, Nillable, Sort, Update
	<b>Description</b> Renewal term in months. This value must match your Salesforce contract, unless you've been granted override permissions.
<b>Label</b> Service Order	<b>Type</b> reference
<b>Name</b> Partner_Order__c	<b>Properties</b> Create, Filter, Group, Sort
	<b>Description</b> Lookup to the related service order record.
<b>Label</b> SFDC Invoice Description	<b>Type</b> string
<b>Name</b> Product_Line_Description__c	<b>Properties</b> Create, Filter, Group, Nillable, Sort, Update
	<b>Description</b> Contains additional invoice details for the product or order. This field is optional.
<b>Label</b> Total Quantity	<b>Type</b> double
<b>Name</b> Quantity__c	<b>Properties</b> Create, Filter, Nillable, Sort, Update
	<b>Description</b> 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
<code>sendOrder</code>	ID	Submit an order with a single ID immediately.
<code>sendOrder</code>	Set of IDs	Submit an order with a set of IDs immediately.
<code>sendOrderAsync</code>	ID	Submit an order with a single ID asynchronously ( <code>@future</code> ).
<code>sendOrderAsync</code>	Set of IDs	Submit an order with a set of IDs asynchronously ( <code>@future</code> ).

## 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 11 Managing Licenses for Managed Packages

### In this chapter ...

- [Get Started with the License Management App](#)
- [Lead and License Records in the License Management App](#)
- [Modify a License Record](#)
- [Refresh Licenses for a Managed Package](#)
- [Extending the License Management App](#)
- [Move the License Management App to Another Salesforce Org](#)
- [Troubleshoot the License Management App](#)
- [Best Practices for the License Management App](#)
- [Troubleshoot Subscriber Issues](#)


Use the License Management App (LMA) to manage leads and licenses for your AppExchange solutions. By integrating the LMA into your sales and marketing processes, you can better engage with prospects, retain existing customers, and grow your ISV business. The LMA is a managed package that is installed in all partner business orgs (PBO) and includes custom objects that track details on packages, package versions, and licenses.

I need to...	Permissions	For details, see...
Configure the LMA	System Admin profile	<a href="#">Get Started with the License Management App</a> on page 308
Bill subscribers or monitor license expiration	Object Permissions: Read	<a href="#">Lead and License Records in the LMA</a> on page 311
Convert trial subscriptions into paying customers	Object Permissions: Edit	<a href="#">Modify a License Record</a> on page 311
Customize the LMO	Object Permissions: Edit	<a href="#">Extend the LMA</a> on page 312
Provision licenses to a subscriber	Object Permissions: Edit	<a href="#">Modify a License Record</a> on page 311
Support subscribers with technical issues	Various permissions (see <a href="#">Assign Permissions to the Subscriber Support Console</a> on page 310 for details)	<a href="#">Support Your AppExchange Customers</a> on page 344

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

 **Note:** The LMA is available only in English.

The LMA is available to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, visit <https://partners.salesforce.com>.

## Get Started with the License Management App

To start managing leads and licenses with the License Management App (LMA), complete these installation and configuration steps.

### [Install the License Management App](#)

The License Management App (LMA) is a managed package that is installed in all partner business orgs. The org that the LMA is installed in is called the License Management Org (LMO).

### [Associate a Package with the License Management App](#)

To receive lead and license records for your package, you connect your License Management Org (LMO), your package, and AppExchange Publishing Console.

### [Configure Permissions for the License Management App](#)

Determine who needs access to the LMA, and set object permissions. Consider using a permission set to assign user permissions.

#### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

## Install the License Management App

The License Management App (LMA) is a managed package that is installed in all partner business orgs. The org that the LMA is installed in is called the License Management Org (LMO).

We strongly recommend that you use your partner business org (PBO) as your LMO. However, you can choose to install the LMA in another production org. Consider installing the LMA in an org that your company is already using to manage sales, billing, and marketing.

Commercial use of the LMA is prohibited in Developer and Partner Developer Edition orgs. Installing the LMA in a Developer Edition org is allowed only if you're building integrations with the LMA and need an environment only for development and testing purposes. You can install the LMA in Enterprise, Unlimited, or Performance Edition production orgs.



**Note:** To confirm whether your PBO already has the LMA installed, skip to step 4.

1. To install the LMA in an org other than your PBO, navigate to the [Partner Community](#).
  - a. From the Partner Community, select the **Support** tab.
  - b. Select **New Case**.
  - c. Select **License Management Application**, and then click **Create a Case**.
  - d. For Subtopic, select **LMA Request**.
  - e. Enter the required information in the Description field, and then click **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 go to the installation URL included in the email.
3. Choose which users can access the LMA, and then click **Install**.
4. To confirm that the LMA is installed, open the App Launcher. If the installation was successful, the License Management App appears in the list of available apps.

#### USER PERMISSIONS

To install packages:

- Download AppExchange Packages

## Associate a Package with the License Management App

To receive lead and license records for your package, you connect your License Management Org (LMO), your package, and AppExchange Publishing Console.

A single LMO can manage multiple 1GP and 2GP packages, but a package can be associated with only one LMO.

1. Connect your packaging org (for 1GP) or your Dev Hub org (for 2GP) to the publishing console on the Partner Community.
  - a. Log in to the [Partner Community](#), and select the **Publishing** tab.
  - a. Select the Organizations tab, and click **Connect Org**.
  - a. Enter the login credentials, and then click **Submit**.

For 1GP packages, enter the login credentials for the packaging org. Repeat this step for all your 1GP packages.

For 2GP packages, enter the login credentials for the Dev Hub org. When you connect the Dev Hub org, all the 2GP packages owned by the Dev Hub org are linked to the publishing console.

2. Select the **Packages** tab.
3. Locate the package you want to link to the LMO, and then click **Register Package**. To register each package you own, repeat this step.
4. Set the default behavior you want for your package license, and then click **Save**.

You can view which packages are linked on the Packages tab in the LMA.

 **Note:** Beta package versions don't display in the LMA. Only managed-released package versions (1GP) and promoted package versions (2GP) are visible in the LMA.

USER PERMISSIONS

To manage licenses in the Partner Community:

- Manage Listings

## Configure Permissions for the License Management App

Determine who needs access to the LMA, and set object permissions. Consider using a permission set to assign user permissions. Ensure that you have the LMA installed, and that you've linked your package to the LMO and AppExchange Publishing Console.

1. Set object permissions for the license, package, and package version custom objects.

Custom Object	Object Permissions
License	To view license records: Assign READ permissions  To modify license records: Assign READ and EDIT permissions
Package	To view package records: Assign READ permissions  To modify package records: Assign READ and EDIT permissions

Custom Object	Object Permissions
Package Version	To view package version records: Assign READ permissions We recommend leaving all package version records as read-only.

- Set field-level security in user profiles or permission sets.

Custom Object	Field-Level Permissions
License	Make all fields read-only.
Package	Make all fields read-only.
Package Version	Make all fields read-only.

- Add related lists to page layouts.

To enable...	Add the Licenses related list to the...
License managers to view the licenses associated with a particular lead	Lead page layout
LMA users to view the licenses associated with a particular account	Account page layout
LMA users to view the licenses associated with a particular contact	Contact page layout

### [Assign Permissions to the Subscriber Support Console](#)

Create a permission set to provide users access to the Subscriber Support Console.

## Assign Permissions to the Subscriber Support Console

Create a permission set to provide users access to the Subscriber Support Console.



**Note:** If you've already assigned these permissions via a profile or another permission set, you can skip this task.

- From Setup, in the Quick Find box, enter *Permission Sets*, and select **Permission Sets**.
- Click **New** and enter your permission set information.
- On the Permission Set Overview page, locate the Apps section, and select **Visualforce Page Access**.
  - Click **Edit**.
  - Add **sfLma.LoginToPartnerBT** and **sfLma.SubscriberSupport** to the list of Enabled Visualforce pages, and then click **Save**.
- On the Permission Set Overview page, locate the System section, and select **System Permissions**. Click **Edit**.

- a. Select **Log in to Subscriber Organization**, and click **Save**.
5. From Setup, in the Quick Find box, enter *Profiles*, and select **Profiles**.
  - a. Click **Edit**.
  - b. Under Custom App Settings, select **License Management App**.
  - c. Under Custom Tab Settings, locate the Subscribers tab and select **Default On**.
  - d. Click **Save**.

## Lead and License Records in the License Management App

Each time a customer installs your managed package, the License Management App (LMA) creates lead and license records.

The key objects in the LMA are Package, Lead, and License.

- **Package**—The LMA includes a Package custom object and a Package Version custom object. These objects display details about each 1GP or 2GP package and package version you've listed on AppExchange.
- **Lead**—The Lead standard object gives you details about who installed your package, such as the installer's name, company, and email address. Lead records created by the LMA are just like the ones you use elsewhere in Salesforce, except the lead source is Package Installation. You can manually convert leads into accounts and contacts. When you convert a lead, the license record links to the converted account or contact.
- **License**—The License custom object gives you control over how many users in the customer's org can access your package and for how long. Each license record links to a lead record and a package record.

To understand which actions you must take and which actions the LMA handles for you, review this table.

Action	Who Takes This Step
Your package is installed by a new subscriber.	Customer or prospect
A lead record is created with the customer's name, company, and email address.	LMA
A license record is created according to the values you specified when you registered the package.	LMA
The lead record is converted to account and contact records. (Optional)	You (ISV partner)
Account and contact records are associated with the license record.	LMA


## Modify a License Record

You can change a customer's access to your offering by modifying a license record using the License Management App (LMA). For example, you can increase or decrease the number of seats included with a license or change the expiration date.

1. In the LMA, locate the license.
2. Click **Modify License**.

When the LMA is installed, the Edit button doesn't appear on the license page layout, and the Modify License button is included instead. This setup is intentional. Only edit license records on the Modify License page. Don't attempt to edit license records programmatically via Apex classes, triggers, or the API.

3. Update the field values as needed.

Field	Description
Expiration	Enter the last day that the customer can access your package, or select <b>Does not expire</b> .
Seats	Enter the number of licensed seats, or select <b>Site License</b> to make your package available to all users in the customer's org. You can allocate up to 99,000,000 seats.
Status	<p>Select a value from the dropdown.</p> <ul style="list-style-type: none"> <li>• <b>Trial</b>—Lets the customer try your offering for up to 90 days. After the trial license converts to an active license, it can't return to a trial state.</li> <li>• <b>Active</b>—Lets the customer use your package according to the license agreement.</li> <li>• <b>Suspended</b>—Prohibits the customer from accessing your offering.</li> </ul> <p> <b>Note:</b> When your offering is uninstalled, its status is set to Uninstalled, and the license can't be edited.</p>

4. Click **Save**.

## Refresh Licenses for a Managed Package

To sync all license records for a package across all subscriber installations, you refresh the license. Refreshing the license can also resolve discrepancies between the number of licenses in a subscriber's org and the number displayed in the License Management App (LMA). Refreshing is required when you move the LMA to a different org.

 **Note:** For each package, you can refresh licenses only one time per week.

1. From the LMA, select the **Packages** tab.
2. Open the package record.
3. Click **Refresh Licenses**. In Lightning Experience, Refresh Licenses is located in the dropdown menu.

## Extending the License Management App

The License Management App (LMA) is a managed package that you can customize and extend. In addition to using the LMA to manage leads and licenses, many partners also integrate it into their existing business processes.

The LMA includes these custom objects:

- [License](#) on page 313
- [Package](#) on page 313
- [Package Version](#) on page 313

You can add custom fields to the objects as long as you don't mark your custom fields as required.

### [Package and Package Version Object Fields](#)

The License Management App (LMA) includes a Package custom object and a Package Version custom object. These objects display details about each 1GP or 2GP package and package version you've listed on AppExchange.

### License Object Fields

Use the License custom object to set limits on how many users in the subscriber's org can use your app and for how long.


### Adding Custom Automation to License Management App Objects

Here are some examples of how you can use the License Management App (LMA) to grow your business and retain customers.

## Package and Package Version Object Fields

The License Management App (LMA) includes a Package custom object and a Package Version custom object. These objects display details about each 1GP or 2GP package and package version you've listed on AppExchange.

To view details about a package record, from the LMA, select the **Packages** tab, and then select the package name. You can view package versions in the Package Version related list.

 **Note:** The LMA creates the package records, which contain critical information for tracking your licenses and packages. Treat these fields as read-only and ensure that your object permissions protect package records.

Package Custom Object Fields	Description
Developer Name	The name of the org that owns the package. For 1GP, the org name is the packaging org. For 2GP, it's the Dev Hub org.
Developer Org ID	The 18-character ID of the org that owns the package. For 1GP, the org ID is the packaging org ID. For 2GP, it's the Dev Hub org ID.
Last License Refresh	The date when the License Refresh tool was last run.
Latest Version	The most recent package version you've released.
Lead Manager	The owner of the lead records that the LMA creates when a customer installs your package.
Next Available Refresh	The date when the License Refresh tool can be run again.
Owner	The LMA owns all package records.
Package ID	The 18-character ID that identifies the package. This ID starts with 033.
Package Name	The name you specified when you created the package.

Package Version Object Fields	Description
Package	The package name and links to the package record's detail page.
Package Version Name	The name you specified when you created the package version.
Release Date	The date you created this package version.
Version Number	The version number in major.minor.patch format. For example, 3.1.0.
Version ID	The 18-character ID of this package version.

## License Object Fields

Use the License custom object to set limits on how many users in the subscriber's org can use your app and for how long.

The License Management App (LMA) creates a license record every time your package is installed in an org. For example, if a subscriber installs two of your 1GP packages and three of your 2GP packages, you have five license records for that subscriber in your LMA. If you deliver a 2GP app that is composed of multiple packages, a unique license record is created for each package in the app. You can allocate up to 99,000,000 seats per subscriber license.

To view details about a license record, select the **Licenses** tab in the LMA, and then select and open the license record.

License records are automatically created and contain critical information for tracking licenses. Do not directly edit the license record. Instead, use the [Modify License](#) on page 311 tool to change the expiration date, license status, and the number of licensed seats.

License Custom Object Fields	Description
Account	A lookup field to the account record for a converted lead.
Contact	A lookup field to the contact record for a converted lead.
Created By	License records are always created by the LMA.
Expiration Date	Displays the expiration date or <code>Does not expire</code> (default).
Install Date	The date the subscriber installed this package version.
Instance	The Salesforce instance where the subscriber's org resides.
Lead	The lead record that the LMA created when the package 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.
License Name	An auto-generated number that represents an instance of a license. License names are in the format of L-00001, and each new license is incremented by one.
Licensed Seats	Displays the number of licenses or <code>Site License</code> (default).
License Status	The type of license: Active, Suspended, Trial, or Uninstalled.
License Type	This is a legacy field and can be ignored.
Org Edition	The edition of the subscriber's org.
Org Expiration Date	Applies only if the subscriber installs your package in a trial org. Indicates the date when the trial org expires. It isn't related to the package license expiration.
Org Status	The status of the subscriber's org: Active, Free, or Trial.
Owner	The LMA owns all license records. Don't edit this field.
Package Version	A lookup field that links to the package version associated with this license.
Package Version Number	The version number in major.minor.patch format. For example, 3.1.0.
Sandbox	Indicates whether the license is for a package installed in a sandbox org.
Subscriber Org ID	The 15-character ID representing the subscriber's org.
Used Licenses	Displays the number of users who have a license to the package.  This field is blank if: <ul style="list-style-type: none"> <li>• A customer uninstalled the package.</li> </ul>

**License Custom Object Fields****Description**

- `Licensed Seats` is set to Site License.

## Adding Custom Automation to License Management App Objects

Here are some examples of how you can use the License Management App (LMA) to grow your business and retain customers.

**EDITIONS**

Available in: both Salesforce Classic and Lightning Experience

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

### Alert Sales Reps Before a License Expires

If you're managing licenses for several packages, it can be difficult to track the various expirations. 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.

## Move the License Management App to Another Salesforce Org

You can move an LMA to a different org, but your package and license records don't automatically move with it. You must manually relink your packages and refresh the licenses.

1. To remove the association between the LMA and the org where it's currently installed, log a case with the Partner Community.
  - 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 click **Submit Case**.
2. [Install the LMA in the new org](#) on page 308.
3. [Associate your packages with the new org](#) on page 309.

**USER PERMISSIONS**

To install packages:

- Download AppExchange Packages

To manage licenses in the Partner Community:

- Manage Listings

4. [Refresh licenses for your packages](#) on page 312.

## Troubleshoot the License Management App

If you're experiencing issues with the License Management App, review these troubleshooting tips.

### [Leads and Licenses Aren't Being Created in the License Management App](#)

When a customer installs your package, leads and license records are created. If these records aren't being created, review these configurations in the License Management Org (LMO). If you resolve your issue using one of these recommendations, your missing licenses appear in the LMA within a few days.

### [Proxy User Has Deactivated Message in the LMA](#)

If you're editing a license and see a "proxy user has deactivated" message, check whether the subscriber org is locked, deleted, or disabled.

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

## Leads and Licenses Aren't Being Created in the License Management App

When a customer installs your package, leads and license records are created. If these records aren't being created, review these configurations in the License Management Org (LMO). If you resolve your issue using one of these recommendations, your missing licenses appear in the LMA within a few days.

### Did the customer complete the package installation?

When a customer clicks **Get it Now** on your AppExchange listing, Salesforce counts this selection as an installation. However, the customer can cancel the installation before it's completed, or the installation could have failed. If the installation doesn't finish, a license isn't created.

### Is State and Country picklist validation enabled in the LMO?

If it's enabled, try disabling it. When this feature is enabled, a known issue prevents leads from being created. 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. 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 a 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're no longer leads.

### Are you using standard duplicate rules for leads?

When a customer installs your package, the LMA checks for existing leads and contacts. If an existing contact matches the customer who installed your package, a lead record isn't created. To complete these checks, the LMA applies [standard lead duplicate rules](#)

and [matching rules](#). If you prefer to have the LMA associate every license with a lead regardless of whether there's an existing contact match, [customize the standard duplicate rule for leads](#) and remove the matching rule for contacts.

## Proxy User Has Deactivated Message in the LMA

If you're editing a license and see a "proxy user has deactivated" message, check whether the subscriber org is locked, deleted, or disabled.

- If the org has been deleted, delete the corresponding license record.
- If the org is locked or if the package has been uninstalled, license records can't be updated.

## Best Practices for the License Management App

---

Follow these best practices when you use the License Management App (LMA).


- To take advantage of entitlements that are unique to AppExchange partners, use your partner business org as your License Management Org.
- Set up My Domain in the Salesforce org where the LMA is installed. A My 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 Salesforce 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 [Apex Reference Guide](#).
- Treat the LMA custom objects as read-only. Use the Modify License page to edit licenses. Don't attempt to directly or programmatically edit license records.
- The LMA automatically creates package, package version, and license records. Customizations, such as adding required custom fields or creating workflow rules, triggers, or validation rules that require custom fields, can prevent the LMA from working properly.

## Troubleshoot Subscriber Issues

---

Use the Subscriber Support Console to access information about your subscribers. Subscribers can also grant you login access to troubleshoot issues directly within your app. After you're granted access, you can log in to the subscriber's org and view their configuration and data to troubleshoot and resolve issues.

To access the Subscriber Overview page, click the organization's name from the **Subscribers** tab in the LMA.

 **Note:** This feature is available to eligible Salesforce partners. For more information on the Partner Program, including eligibility requirements, see [www.salesforce.com/partners](https://www.salesforce.com/partners).

### [Request Login Access from Subscribers](#)

To log in to a subscriber org, first request login access from the subscriber.

### [Log In to Subscriber Orgs](#)

After your subscriber has granted you login access, you can log in to the subscriber org to troubleshoot the issue. We recommend logging in using multi-factor authentication (MFA).

### [Debug Subscriber Orgs](#)


After logging in to a subscriber's org, you can view logs, obfuscated code in your package, and initiate ISV Customer Debugger sessions.

## Request Login Access from Subscribers

To log in to a subscriber org, first request login access from the subscriber.

Ask the subscriber to enable either **Grant Account Login Access** or **Grant Login Access**. If they don't see your company 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. In this scenario, only an admin with the Manage Users permission can grant access.
- The org setting **Administrators Can Log in as Any User** is enabled.

 **Note:** When the org setting **Administrators Can Log in as Any User** is disabled, login 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 subscriber org's audit trail.

## Log In to Subscriber Orgs


After your subscriber has granted you login access, you can log in to the subscriber org to troubleshoot the issue. We recommend logging in using multi-factor authentication (MFA).

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

### USER PERMISSIONS

To log in to subscriber orgs:

- Log in to Subscriber Org

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

## Multi-factor Authentication Requirement to Log In to a Subscriber Org


In Spring '22, the Require Multi-Factor Authentication For Logins to Subscriber Orgs release update will be enforced on all License Management Orgs (LMO). This update provides subscribers an extra layer of security by verifying the identity of the user accessing their org. You also have more control over which users log in to a subscriber org.

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

## Log In to a Subscriber Org

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

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

 **Note:** Some subscribers require MFA in addition to the MFA required for the LMO. Ask your subscriber if their org requires MFA to log in. If so, your login attempt sends an MFA notification to your subscriber, and your login is blocked until your subscriber

responds to the notification. To ensure that your subscriber is available to respond to the MFA notification, consider coordinating a specific login time.

## Best Practices for Logging In

- Create an audit trail that indicates when and why a subscriber org login has occurred. You can create an audit trail by logging a case in your LMO before each subscriber org login.
- When you access a subscriber org, you're logged out of your LMO. You can set up a My Domain to not be automatically logged out of your LMO when you log in to a subscriber org. To set up a My Domain subdomain, from Setup, in the Quick Find box, enter *My Domain*, then select **My Domain**.
- Allow only trusted support and engineering personnel to log in to a subscriber's org. Because 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 login access by giving the Log in to Subscriber Org user permission to specific support personnel via a profile or permission set. See [Assign Permissions to the Subscriber Org Console](#) on page 310.

## Debug Subscriber Orgs

After logging in to a subscriber's org, you can view logs, obfuscated code in your package, and initiate ISV Customer Debugger sessions.

### Troubleshoot with Debug Logs

You can debug your code by generating Apex debug logs that contain the output from your managed package. 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, in the Quick Find box, enter *Debug Logs*, and 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 because they contain your unobfuscated Apex code.

You can also view and edit data contained in protected custom settings from your managed packages when logged in as a user.

### Troubleshoot with the ISV Debugger

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

For details, see the [ISV Customer Debugger](#) documentation.

SEE ALSO:

[Salesforce Help: Open the Developer Console](#)

## CHAPTER 12 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 org 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 13 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](#)
- [Update Your Trial](#)
- [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 has several parts: the Trialforce management organization, Trialforce source organizations, and Trialforce templates. Before you set up Trialforce, learn how these parts work together to deliver a trial of your AppExchange solution.

### **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 log a case in the [Salesforce Partner Community](#) to request a TMO. The two tasks you perform in the TMO are:

- Create Trialforce source organizations.
- Define templates for custom branding.

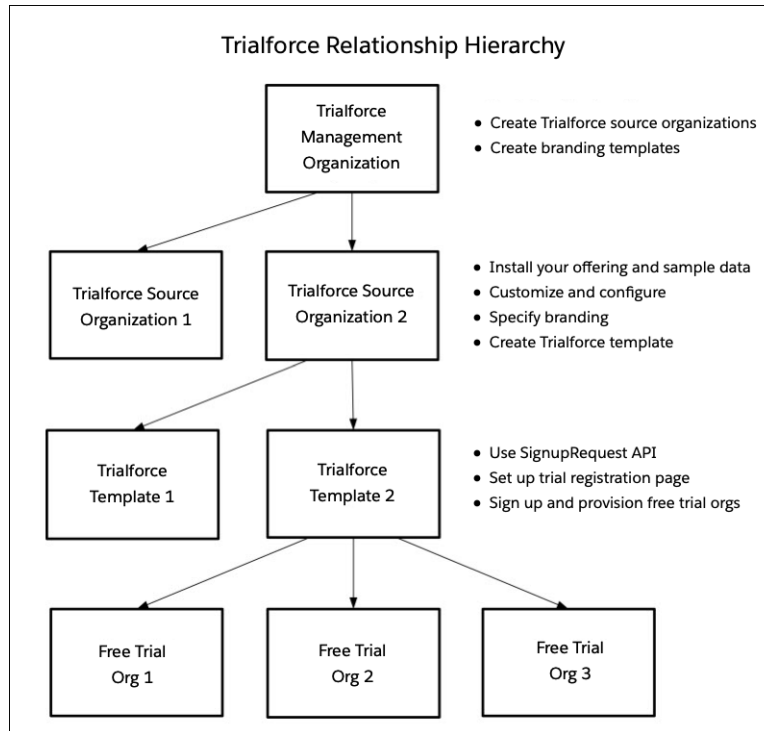
### **Trialforce Source Organization (TSO)**

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.
- Configure the TSO to be exactly as you want your customers to experience it.
- Specify branding by choosing from the templates you previously created in the TMO.
- Create a Trialforce template, which becomes the basis for all trial orgs.

### **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. You specify a Trialforce template when you generate a trial org using the SignupRequest API and when you create a demo org from the Environment Hub. The template defines the trial org that is provisioned each time a customer signs up for a trial of your offering.



The TMO, TSOs, and Trialforce templates have a hierarchical relationship.

- 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, you release a new version of your component after you've started using Trialforce. You can install the updated version into the previous TSO and generate a new Trialforce template from it. To request a trial for the new version, use the SignupRequest API with the new Trialforce template ID. All trial orgs created using the new template ID 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 offering, and one Trialforce template for each version or edition of your offering. Splitting up the configuration process across these different levels makes it easier to maintain and update your trials. Each time you change something, such as the version, its branding, or a configuration detail of the trial org, 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 offerings, versions, and editions.

After you've configured a TMO, TSO, and Trialforce template, choose how to provide trials to prospective customers.

- **Use AppExchange:** Customers begin a trial of your offering directly from an AppExchange listing. This is the quickest, easiest way to make a trial available because it requires only a few steps to configure.
- **Use the API:** You provision a trial of your offering programmatically using the SignupRequest API. This approach is ideal if you want 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.

 **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](#)

### EDITIONS

Available in: Salesforce Classic

Available in: **Developer Edition**

### USER PERMISSIONS

To manage Trialforce:

- [Customize Application](#)

## 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**.

## 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.

 **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.

### EDITIONS

Available in: Salesforce Classic

Available in: **Developer Edition**

### USER PERMISSIONS

To manage Trialforce:

- Customize Application

### EDITIONS

Available in: Salesforce Classic

Available in: **Developer Edition**

### USER PERMISSIONS

To manage Trialforce:


- Customize Application

## 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.

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

### 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 company-specific My Domain name. 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.

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 your My 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

### EDITIONS

Available in: Salesforce Classic

Available in: **Developer Edition**

### USER PERMISSIONS

To manage Trialforce:

- Customize Application

- 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.

 **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.

### 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

**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](https://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 drive 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](#).

1. Log in to the [Salesforce 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**.

To ensure the test drive user is granted read-only access, enter credentials for the *evaluation* user. Don't enter admin credentials as this gives the user read-write access.

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](#)

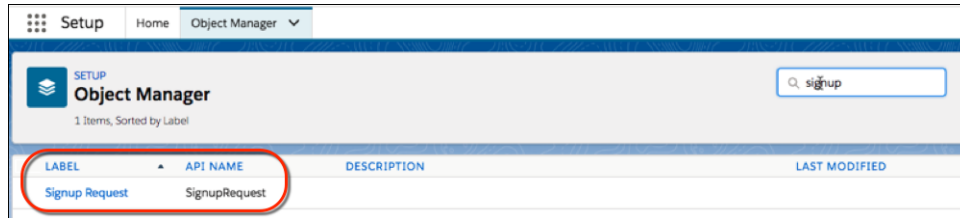
Use Trialforce to provision a free trial of your solution for prospective customers.

## 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 Experience Cloud site
- 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.

## Update Your Trial

If you update your offering or change custom branding, update your trials to reflect the changes.

To update a trial, you must first:

- 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 use a new one. If you use a new TSO, be sure to link it to AppExchange.

Then complete the following steps.

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 changing custom branding.
3. Create a Trialforce template for your trial.
4. [Submit the template for review.](#)
5. Get the template approved for SignupRequest API use.
  - a. Log in to the [Salesforce Partner Community](#).
  - b. Click **Support**.
  - c. Click **New Case**.
  - d. Hover over the **AppExchange (ISV) App Setup & Management > Trialforce** category.
  - e. Click **Create a Case**.
  - f. Provide a case subject, and select a severity level.
  - g. For Subtopic, select **Signup API Feature Request**.
  - h. For Description provide the TSO ID, the new Trialforce template ID, and the org to use for creating sign ups.
  - i. Click **Submit Case**.

## 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. Trial orgs are a fully functioning Salesforce organization. Your customer will have your app installed and can later install other 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 14 Support Your AppExchange Customers

In this chapter ...


- [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.

## Usage Metrics

 **Note:** Usage Metrics is scheduled for retirement in all Salesforce orgs as of the Winter '22 Release. Switch now to [AppExchange App Analytics](#), which also provides usage data about how subscribers interact with your AppExchange solutions. You can use these details to identify attrition risks, inform feature development decisions, and improve user experience. To enable AppExchange App Analytics on your security-reviewed managed packages, follow the instructions in [Request AppExchange App Analytics](#).

### EDITIONS

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

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.

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](https://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/gml2__f1","1","1","66.0"
"00Dxx0000001gbk","org1","Enterprise Edition","TRIAL","1.0","/apex/gml2__f2","1","1","128.0"
"00Dxx0000001gbk","org1","Enterprise Edition","TRIAL","1.0","/apex/gml2__f3","1","1","107.0"
"00Dxx0000001gbf","org1","Enterprise Edition","TRIAL","1.0","/apex/gml2__f1","5","1","73.6"
"00Dxx0000001gbf","org1","Enterprise Edition","TRIAL","1.0","/apex/gml2__f2","1","1","72.0"
"00Dxx0000001gbf","org1","Enterprise Edition","TRIAL","1.0","/apex/gml2__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.

## Supported Calls

`query()`, `delete()`

## Fields

Field Name	Details
MetricsDataFile	<b>Type</b> base64  <b>Properties</b> Filter, Query, Sort  <b>Description</b> A text file containing the usage data encoded in Base 64.
MetricsDataFileContentType	<b>Type</b> string  <b>Properties</b> Filter, Query, Sort  <b>Description</b> The format of the data file. Currently, the only allowed value is <code>text/csv</code> .
MetricsDataFileLength	<b>Type</b> int  <b>Properties</b> Filter, Query, Sort

Field Name	Details
	<b>Description</b> The size of the data file in bytes.
MetricsRunDate	<b>Type</b> dateTime <b>Properties</b> Filter, Query, Sort <b>Description</b> The date when the usage metrics collection job was run.
MetricsEndDate	<b>Type</b> dateTime <b>Properties</b> Filter, Query, Sort <b>Description</b> The end time and date for the data collection.
MetricsStartDate	<b>Type</b> dateTime <b>Properties</b> Filter, Query, Sort <b>Description</b> The start time and date for the data collection.
MetricsType	<b>Type</b> picklist <b>Properties</b> Filter, Query, Sort <b>Description</b> The type of data being collected. The possible values are CustomObject and Visualforce.
NamespacePrefix	<b>Type</b> string <b>Properties</b> Filter, Query, Sort <b>Description</b> The namespace prefix of the package for which data is being collected.
SendingInstance	<b>Type</b> string

Field Name	Details
	<b>Properties</b> Filter, Query, Sort
	<b>Description</b> 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.

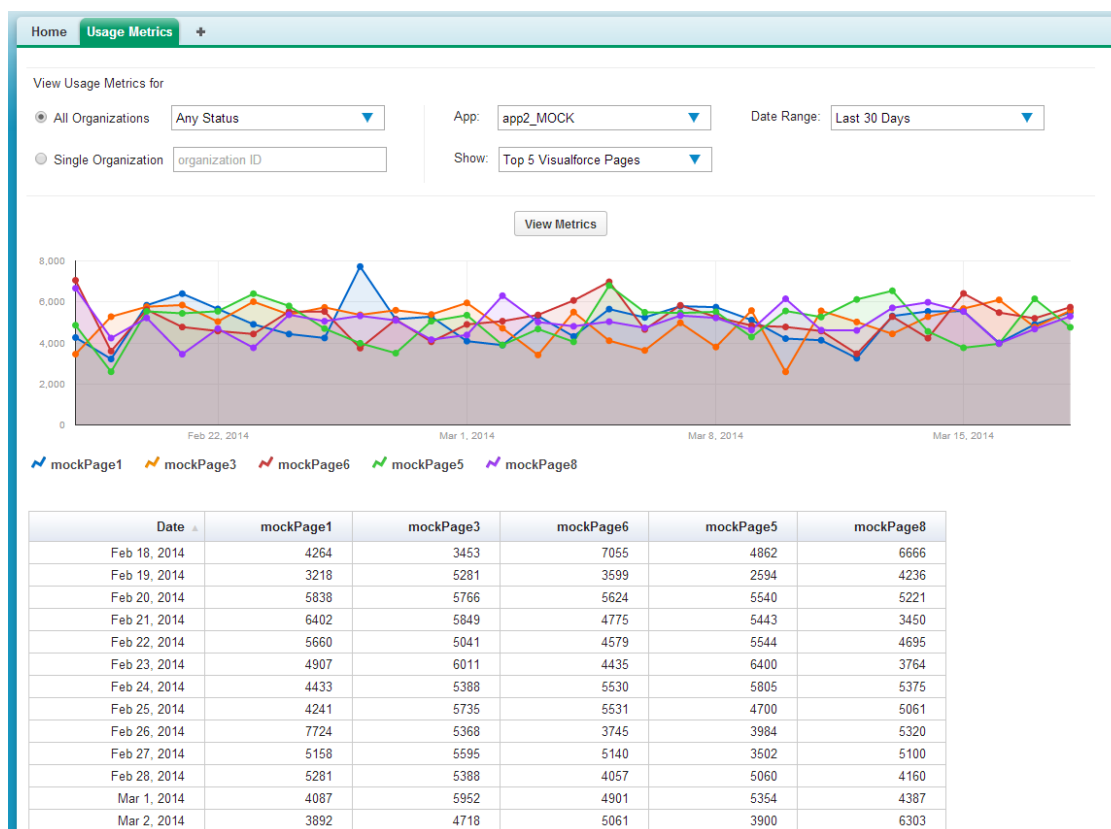
The app is intended as a reference implementation, for illustration purposes only. It's distributed as an unmanaged package, so you can review its components and extend or customize it to meet your requirements. If your visualization needs are more complex, you can export the raw metrics data from the 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.



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 15 Update Your Solution

### In this chapter ...

- [About Package Versions](#)
- [Create and Upload Patches](#)
- [Working with Patch Versions](#)
- [Publish Upgrades to Managed Packages](#)
- [Push Package Upgrades to Subscribers](#)


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.

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.

### 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

## Create and Upload Patches

---

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

---

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.

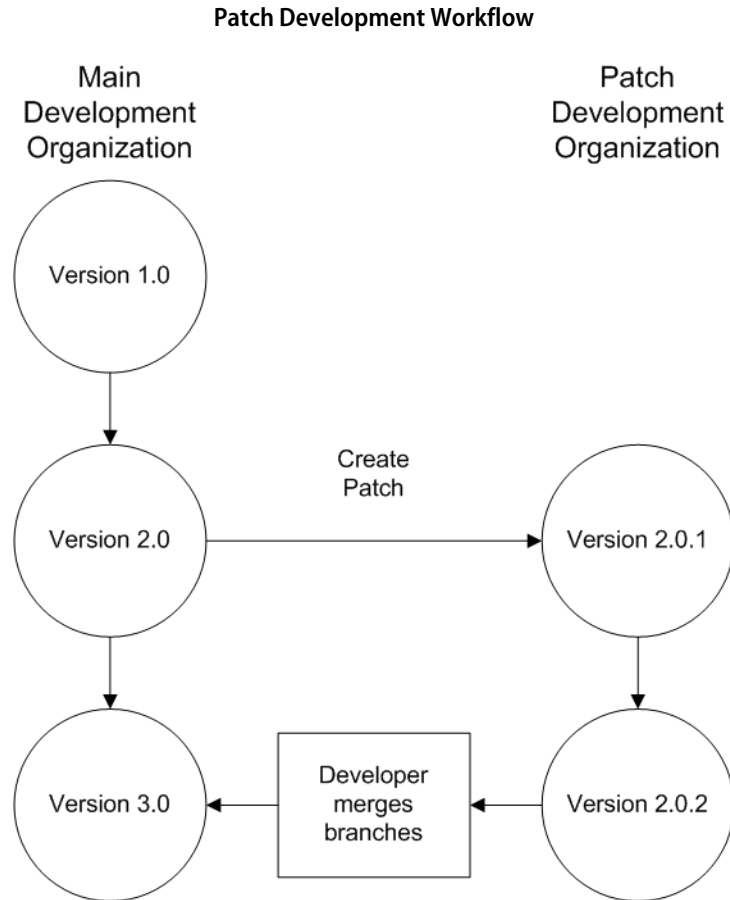
## Patch Development Organizations

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:



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.

The best way to keep track of your package versions with Git source control. Learn about Git from this trail: <https://trailhead.salesforce.com/en/content/learn/modules/git-and-git-hub-basics>.

Version control is integrated into Visual Studio Code. See [Salesforce Extensions for Visual Studio Code](#) and [Version Control in Visual Studio Code](#) for details.

## 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, <code>PackageDevClass</code> , containing a global method <code>m1</code> .		
Upload as Managed - Released version 1.0 of a package that contains <code>PackageDevClass</code> .	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, <code>SubscriberClass</code> , that references <code>m1</code> in <code>PackageDevClass</code> .	
Deprecate <code>m1</code> and create a new method, <code>m2</code> .		
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

Package Developer Action	Subscriber Action	Notes
		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.

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

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

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

5. When you receive an email with the link to the upload on Salesforce AppExchange, notify your installed users that the new version is ready. Use the list of installed users from the License Management Application (LMA) to distribute this information. The License Management Application (LMA) automatically stores the version number that your installers have in their organizations.

### 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 from First-Generation Managed Packages

After you've uploaded a  Managed - Released first-generation managed package, you sometimes want to delete a component from your org. The options available for deleting a component depends on the situation in which you are trying to delete the component.

Let's say one of the following situations occurs:

- The component, after being 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.
- 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, use a two-stage process to delete Visualforce pages, global Visualforce components, and global Lightning components from a managed package. When you upgrade a package in a subscriber org, the Visualforce pages, global Visualforce components, or Lightning 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.

Perform the deletion steps in this order if you're deleting the following types of components:

- A Visualforce page or global Visualforce component that refers to or uses public Apex classes or public Visualforce components.
- An Aura component with global access
- A Lightning web component with an `isExposed` value of `true`

**1. Stage one: Remove references.**

- i. Edit the global component that you want to delete.
  - For Visualforce: Edit your Visualforce page or global Visualforce component to remove all references to public Apex classes or public Visualforce components.
  - For Lightning components: Edit the global Lightning component to remove all references to other Lightning components.
- 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, global Visualforce component, or global Lightning 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
- Lightning components (Aura and Lightning web components)
- Permission sets
- Record types
- S-Controls
- Static resources
- Validation rules
- Visualforce components
- Visualforce pages

For a more complete list, see [Available Components](#) on page 38.

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:** Educate your customers about the potential impact of deleted components. Consider listing all custom components you've deleted, and specifying any actions needed, in the Release Notes for your upgraded package.

The following restrictions apply when deleting managed components.

- If a component of any type is referenced by any other metadata, such as workflow rules, validation rules, or Apex classes, it is not deletable.
- 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, the component is deleted forever.



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

You can retrieve these types of components.


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

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

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

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

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

Attribute	Description
Action	If the  Managed - Released package hasn't been uploaded with the component deleted, this contains an <b>Undelete</b> 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.

Attribute	Description
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

## Push Package Upgrades to Subscribers

A *push upgrade* is a method of automatically upgrading your customers to a newer version of your package. This feature 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.

Use the Push Upgrade Exclusion List to exclude specific subscriber orgs from a push upgrade. You can specify up to 500 comma-separated org IDs.

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.

## Push Upgrades

### Overview of Push Upgrade 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

- 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.

## Assign Access to New and Changed Features

Determine how to provide existing non-admin users access to new and changed features. By default, any new components included in the push upgrade package version are assigned only to admins.

If the push upgrade includes:	We recommend you:
New features	<p>Notify admins about the changes the upgrade introduces, and ask them to assign permissions to all users of the package.</p> <p>This approach allows admins to choose when to make the new features available.</p>
Enhancements to existing features	<p>Include a post-install script in the package that assigns permissions to the new components or new fields automatically.</p> <p>This approach ensures that current users of the package can continue using features without interruption.</p> <p> <b>Note:</b> Post-install scripts aren't available to unlocked packages.</p>

## Sample Post Install Script for a Push Upgrade

 **Note:** Post-install scripts can be used with first and second-generation managed packages only.

Automate 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 125.

In this sample script, the package upgrade contains new Visualforce pages and a new permission set that grants access to those pages. The script performs the following actions.

- Gets the Id of the Visualforce pages in the old version of the package
- Gets the permission sets that have access to those pages
- Gets the list of profiles associated with those permission sets
- Gets the list of users who have those profiles assigned
- Assigns 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];
List<PermissionSetAssignment> PermissionSetAssignmentList = new
List<PermissionSetAssignment>();
for (User us : UserList) {
    PermissionSetAssignment psa = new PermissionSetAssignment();
    psa.PermissionSetId = PermissionSetToAssign.id;
    psa.AssigneeId = us.id;
    PermissionSetAssignmentList.add(psa);
}
insert PermissionSetAssignmentList;
}
}

```

```

// Test for the post install class
@Test
private class PostInstallClassTest {
    @Test
    public static void test() {
        PostInstallClass myClass = new PostInstallClass();
        Test.testInstall(myClass, null);
    }
}

```

## Scheduling Push Upgrades

After you've created an updated version of your package, you can automatically deploy it to customers using a push upgrade.

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.


### USER PERMISSIONS

To push an upgrade:

- Upload AppExchange Packages

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 355.

### Schedule a Push Upgrade Using the UI

-  **Note:** Only first-generation managed packages can schedule a push upgrade using the UI. To schedule a push upgrade for unlocked and second-generation managed packages, use the [PackagePushRequest](#) object in the SOAP API.

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** dropdown 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.
Organization ID	The ID of the org.
Organization Name	The name of the org. To view the upgrade history for the org, click on the org name.
Primary Contact	The name of the user 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.

-  **Note:** For unlocked and second-generation managed packages, authenticate to your Dev Hub.

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 *SOAP API Developer Guide*.

# 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](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
Content	✓	✓	✓	✓
Experience Cloud site	\$	\$	\$	\$
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	✓	✓		
WDC	\$	\$	\$	\$
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

## ISVforce User License Comparison

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](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:** Starting Spring '16, 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. An ISV Portal license is best used when projected user volumes 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 other 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

\* With the Orders Platform permission set license (PSL), available to OEM partners only, administrators can give users with Lightning Platform user licenses access to Contracts, Products, Price Books, and Orders. Orders functionality is automatically available to all licenses except the 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	✓	✓	✓	✓	✓	✓
Enterprise 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	✓	✓	✓	✓	✓	✓
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 Organization Limits (Added Per User)	OEM Embedded	ISV Portal	ISV Portal with Sharing	Customer Community	Customer Community Plus	Partner Community
Data Storage	20 MB	0	2 MB	0	2 MB per member (member-based license) 1 MB per member (login-based license)	5 MB

## OEM User License Comparison

Additional Organization Limits (Added Per User)	OEM Embedded	ISV Portal	ISV Portal with Sharing	Customer Community	Customer Community Plus	Partner Community
File Storage	2 GB	0	0	0	0	0

For data storage, each OEM Embedded organization is allocated a minimum of 10 GB. For example, an OEM Embedded organization with 20 users at 20 MB per user receives 400 MB plus 10 GB, or 10.4 GB total data storage. An OEM Embedded organization with 100 users receives 12 GB because 100 users multiplied by 20 MB per user is 2 GB.

For file storage, each OEM Embedded organization is allocated a per-user limit multiplied by the number of users in the organization plus a per organization allocation of 11 GB. For example, an OEM Embedded organization 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 Organization	File Storage Minimum per Organization	Storage Allocation Per User License
OEM Embedded	1 GB, plus 5 MB for each Gold Partner license	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	<ul style="list-style-type: none"> <li>Salesforce: 1,000</li> <li>Salesforce Platform: 1,000</li> </ul>	100,000 + (number of licenses x calls per license type)

Limits are enforced against the aggregate of all API calls made to 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:

## 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.

## EDITIONS

Available in: Salesforce Classic ([not available in all orgs](#))

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

### 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

- analytics [210](#), [235–236](#), [253–255](#), [262–263](#)
- Apex
  - behavior in packages [356](#)
  - deprecation effects [357](#)
- API token
  - requesting [176](#)
- app
  - installation options [151](#)
  - patch (version) updates [175](#)
  - search optimization [175](#)
- AppExchange
  - deleting components [361](#)
- AppExchange Checkout [178](#)
- Apps
  - uploading [37](#), [115](#)
- Attributes [34](#)

## B

- Beta packages
  - uninstalling [123](#)
  - uploading [113](#)

## C

- Channel Order App [265](#)
- Checkout [178](#)
- company name [174](#)
- Components [34](#)
- Creating packages [37](#), [115](#)
- Creating patches [354](#)

## D

- data file [346](#)
- Dependencies [72](#)
- Deployment [123](#)
- dev hub [105](#)
- Developer Tools [105](#)
- Developing
  - partner WSDL [83](#)
- Dynamic Apex
  - supporting multiple editions [90](#)

## E

- Editions [3](#)
- Extending packages [89](#)

## F

- Feature Management App
  - considerations [327](#)
  - installation [322](#)
  - known issues [327](#)
  - setup [322](#)
- Feature parameters
  - best practices [326](#)
  - considerations [327](#)
  - de-protecting custom objects [326](#)
  - de-protecting custom permissions [326](#)
  - fields [321](#)
  - known issues [327](#)
  - limits [326](#)
  - LMO-to-subscriber [324–325](#)
  - objects [321](#)
  - protecting custom permissions [326](#)
  - subscriber-to-LMO [325](#)
  - types [321](#)
- feedback [177](#)
- FMA [320–322](#), [324–327](#)
- free trial
  - create [177](#)
  - vs test drive [177](#)

## G

- Group edition
  - access control [87](#)
  - limits [87](#)
  - packages [89](#)
  - using Apex [87](#)

## I

- ideas [177](#)
- industries [174](#)
- Installing packages [120](#)
- introduction [345](#)

## L

- licenses
  - choose settings [152](#)
- listing
  - add categories [166](#)
  - analytics [172](#)
  - delisted by Salesforce [177](#)

Login [317](#)

## M

manage scratch orgs [105](#)

managed package

change [175](#)

change a listing [176](#)

register [152](#)

Managed packages

component availability [123](#)

component behavior [58](#)

group edition [87](#), [89](#)

limits for group edition [87](#)

limits for professional edition [87](#)

professional edition [87](#), [89](#)

push upgrades [363–365](#)

status [34](#)

supporting multiple editions [89–90](#)

upgrading [352](#)

MetricsDataFile object [347](#)

## O

Objects

MetricsDataFile [347](#)

Operational scope [72](#)

## P

Package installation [120](#)

Package versions

behavior versioning Apex [356](#)

deprecating Apex [357](#)

Packages

about [2](#), [34](#)

component availability [123](#)

component behavior [58](#)

creating [37](#), [115](#)

deleting components [361](#)

dependencies [72](#)

designing [18](#)

developing [34](#)

distributing [145](#)

Editions [34](#)

installing packages [91](#)

installing using the API [124](#)

managed [34](#)

managing feature access [320–327](#)

status [34](#)

terminology [34](#)

tracking activation metrics [320–327](#)

Packages (*continued*)

understanding [34](#)

uninstalling using the API [124](#)

unmanaged [34](#)

uploading [37](#), [115](#)

partner account [174](#)

Partner WSDL [83](#)

Patch versions

creating [354](#)

uploading [354](#)

patches [175](#)

Permission set groups [75](#)

Permission sets [73](#)

popularity [176](#)

Professional edition

access control [87](#)

limits [87](#)

packages [89](#)

using Apex [87](#)

Profile settings [73](#)

provider profile

and partner accounts [174](#)

create or edit [149](#)

Push upgrades [363](#), [365](#)

## R

relevance [176](#)

review

edit [175](#)

write [178](#)

## S

Salesforce DX [105](#)

sandbox [175](#)

scratch org allocations [103](#)

search optimization [175](#)

services

search optimization [175](#)

setting up [345](#)

Subscriber support [317](#)

Supporting multiple editions [90](#)

## T

test drive

vs free trial [177](#)

Testing [3](#), [109](#)

trial template

description [177](#)

## Index

Trialforce

Lightning Experience [342](#)

Tutorials [4](#)

## U

Uninstalling packages [123](#)

Unmanaged packages

component behavior [58](#)

Upgrading packages [352](#)

Uploading beta packages [113](#)

Uploading packages [37](#), [115](#)

Uploading patches [354](#)

Usage Metrics [345–346](#)

using an extension package [89](#)

using dynamic Apex [89–90](#)