COMPARING MANAGED PACKAGE GENERATIONS

The changes, similarities, and technical nuances of salesforce's second-generation managed packages.

Post Date:

Author: Bohdan Dovhan

Categories: Salesforce

Keywords: SFDX, CI, 2GMP, ISV, AppExchange, Managed Package, Second Generation, DevOps

softserve

INTRODUCTION

Managed packages are the vehicle that Salesforce partners use to build, share, or sell applications to customers on Salesforce's AppExchange.

Over the last year, Salesforce ushered in a new way for partners to develop, distribute, and manage apps and metadata with a second-generation managed packaging or 2GMP.

This new generation streamlines and simplifies elements such as organizing your source, crafting smaller modular packages, and integrating with your version control system. It's heavily dependent on the Salesforce Developer Experience (SFDX) command-line interface (CLI), meaning your packaging operations can now be handled with these data and metadata management tools. These features can also be automated using scripts, freeing up innovation and iteration by developers to drive even greater business success.

Are you ready to accelerate your developer team's collaboration, enable source-driven development, and reach new heights of business agility? Then read on as we highlight the differences and similarities between the two managed package generations. I'll follow it with a deep dive into some technical nuances and commonly asked questions about 2GMP.

DIFFERENCES BETWEEN FIRST- AND SECOND-GENERATION MANAGED PACKAGES

While some features remain the same, there are crucial differences between the first- and second-generation managed packages.

1. What is the source of truth?

The first key difference is their source of truth.

For first-generation managed packages (1GMP), the source of truth is the packaging organization (known as the “packaging org”) or the patch organization (known as the "patch org").

When a developer signs up for a new developer edition organization, they will start building a package specifically within this packaging org.

Whenever a new package version is uploaded, it collects the source code from the developer edition and embeds it in the package.

For second-generation managed packages (2GMP), there is no patch org or packaging org. Instead, the source of truth for 2GMP is the version control system (VCS). Every component committed and pushed to the VCS is included in the package version. Every component committed and pushed to the VCS is included in the package version.

2. Who is the owner of the package and its metadata?

The answer is simple for 1GMP—the packaging org is the owner of both the package and its metadata.

However, for 2GMP, things aren't as easy.

The metadata here resides in the VCS, but the package itself is owned by the developer hub organization or dev hub org.

This dev hub org is the home of all the packages you want to develop from a single organization. It's also the hub for all the scratch orgs, which are source-driven, temporary, and disposable organizations.

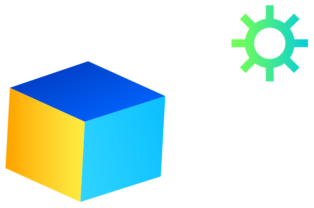
Scratch orgs are useful for continuous integration and continuous delivery (CI/CD). They can be easily created and then discarded with a command-line script, allowing for greater flexibility.

In 1GMP, the packaging org is the owner of both the package and its metadata. For 2GMP, the VCS owns the metadata while the dev hub org owns the package.

3. How many packages may belong to an org?

Only one first-generation managed package may be created in the packaging org.

However, the dev hub can own many second-generation managed packages.



4. Where is the namespace (the prefix used by managed packages to isolate the metadata scope) registered?

For 1GMP, the namespace is registered in the packaging org.

While there's no packaging org concept for 2GMP, there is concept of the namespace org. This is where you first register a namespace. Then you link that namespace org to your dev hub.

5. How many packages can share namespace?

Multiple 2GMPs can share the same namespace, while only a single 1GMP can do the same.

6. What are the options to share code?

The only way to share the code with other packages or subscribers in 1GMP is by using the global access modifier keyword.

While you can still use the global access modifier in 2GMP, you also have another option. You can use the annotation @namespaceAccessible. It will share the code to the packages that use the namespace but hide it from the subscribers, making it more agile.

7. Can package:create or uninstall be automated?

Automation for all operations by SFDX CLI comes standard for 2GMP.

Unfortunately, 1GMP doesn't allow certain operations such as create or uninstall to be automated.

8. Is branching supported in package versioning?

For 2GMP, the answer is easy: yes, it supports branching.

However, 1GMP does not, as its packaging versioning is linear. That means that while you're developing an app, you can release a version 1.0 followed by releasing version 1.1. You can then follow that up with a version 2.0.

However, once you release a version 2.0, you can no longer release a version 1.2 in 1GMP.

9. Can patch versions be created?

In 1GMP, patch versions can be created from a patch org. These are dedicated environments that contain the code and metadata for a specific patch version.

Since there's no patch org concept in 2GMP, their patch versions can be created only from SFDX CLI.

Salesforce released this second generation to eliminate much of the complexity developers previously encountered when working with multiple Salesforce orgs. Making it easier to build apps and packages meant that Salesforce partners could increase innovation.

SIMILARITIES BETWEEN FIRST- AND SECOND-GENERATION MANAGED PACKAGES

Though it's important to highlight these differences and improvements, there remain numerous similarities between first- and second-generation managed packages.

These similarities center around what both generations of managed packages allow an independent software vendor, or ISV, to do.

First, both generations allow ISVs to iterate and create package and patch versions and install and uninstall packages in subscriber orgs.

Next, you can easily list your package on AppExchange, no matter which generation you use. AppExchange is the Salesforce Marketplace where packages can be published by ISVs or found by users for installation.

You can list your package on AppExchange with both generations. You also can submit your package for the AppExchange security review. The Salesforce security team conducts this rigorous four-to-six-week process before listing the product so that customers using AppExchange can be confident that their data is protected.

The third similarity between the 1GMP and 2GMP is that both allow you to utilize the License Management App. This lets you control the licenses available for customers. This feature doesn't apply to apps listed for free or which have an installation fee, but it's a critical tool for others.

With the License Management App, you can track the number of customer installations and the package version installed. One of the most significant benefits is that you're able to manage the leads for each license.

The License Management App benefits your customers as well since it houses the Subscriber Support Console. This feature lets you gain permission from your customer before logging into their org to troubleshoot any issues.

Lastly, both 1GMP and 2GMP support using the Feature Management App. In this app, you're able to define which customers have access to your application's parameters.

All these features available in 1GMP and 2GMP make both generations more usable and easily accessible to meet developer needs.

TECHNICAL NUANCES AND FAQ FOR 2GMP

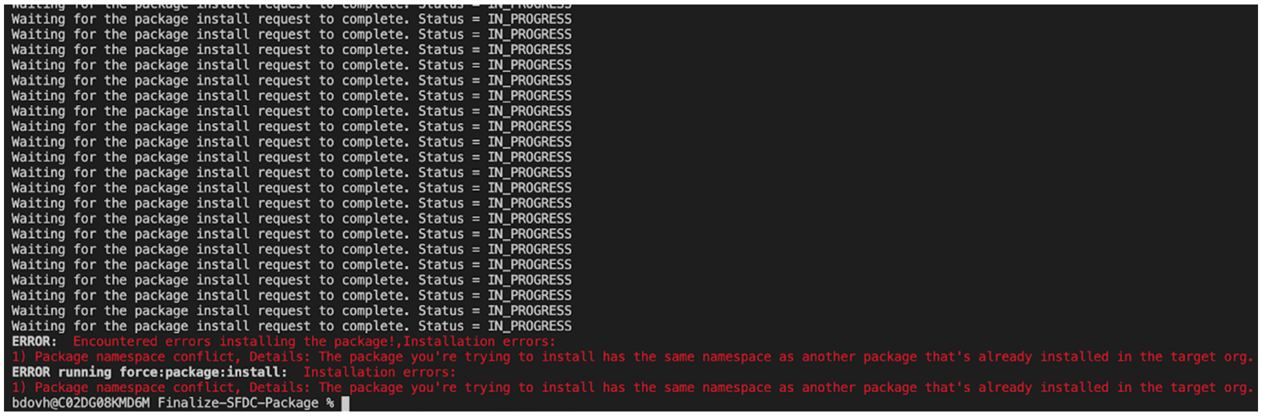
While both generations share differences and similarities, some technical nuances for 2GMP have created developer questions. Let's explore some of them.

1. Can both generations share a namespace?

One of the biggest developer questions—and one that I asked at a conference—is whether 1GMP and 2GMP can share a namespace.

The conference moderators said no, and that is technically true. You cannot install both 1GMP and 2GMP with a shared namespace into the same org,

If you try to do so, it will send an error stating that the package you're trying to install has the same namespace as another package already installed in the target org.



```
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
Waiting for the package install request to complete. Status = IN_PROGRESS
ERROR:  Encountered errors installing the package!,Installation errors:
1) Package namespace conflict, Details: The package you're trying to install has the same namespace as another package that's already installed in the target org.
ERROR running force:package:install:  Installation errors:
1) Package namespace conflict, Details: The package you're trying to install has the same namespace as another package that's already installed in the target org.
bdovh@C02DG08KMD6M Finalize-SFDC-Package %
```

Problem:
1. Package namespace conflict
The package you're trying to install has the same namespace as another package that's already installed in the target org.

But while they both cannot be installed to the same org, the first one can be installed into one org, and another one can installed into another org.

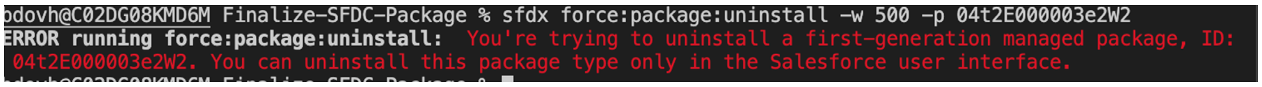
If a developer owns a namespace, some of his or her customers might use a 1GMP with a given namespace, while the others might use 2GMP having the same namespace. However, none of the customers may use both on the same org.

Overall, it is possible for a customer to have multiple orgs and they can use one package on one org and another package on another org.

2. When can you use package install or uninstall commands?

Developers can use the sfdx:force:package:install command to install both 1GP (unmanaged packages) or 1GMP (classic managed packages) into an org. However, you cannot use the sfdx:force:package:uninstall command to uninstall that same 1GP or 1GMP.

If you try to uninstall a first-generation package this way, you will get the following error:



```
bdovh@C02DG08KMD6M Finalize-SFDC-Package % sfdx force:package:uninstall -w 500 -p 04t2E000003e2W2
ERROR running force:package:uninstall:  You're trying to uninstall a first-generation managed package, ID:
 04t2E000003e2W2. You can uninstall this package type only in the Salesforce user interface.
```

Salesforce's documentation states this command will only uninstall a 2GMP or unlocked packages from the target org.

To uninstall a first-generation package, a developer must use the Salesforce user interface.

3. Can you push upgrades?

This feature is meant to push package upgrades to subscribers without their explicit consent.

Many in the industry don't believe it possible to schedule push upgrades for 2GMP. This appears incorrect. While you can't schedule the push upgrade for 2GMP using the Salesforce UI, you can still schedule it using SOAP API.

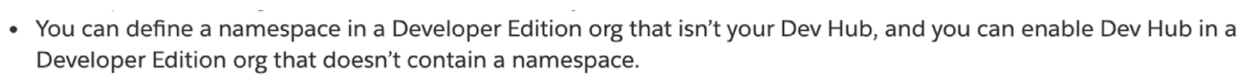
4. Which metadata components does each generation support?

Salesforce provides a helpful cheat sheet that lists which metadata components are supported by the different package generations. It ranges from support by only 1GMP, only 2GMP, by both, or by neither.

5. What about namespace registration and dev hub compatibility?

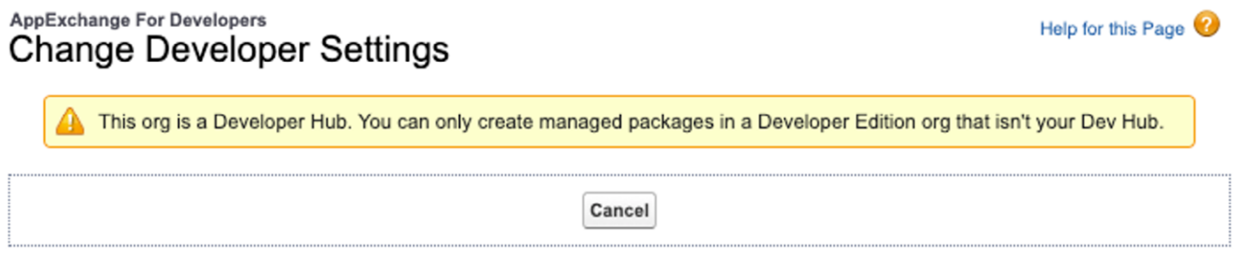
It's important to note that namespace registration and dev hub are incompatible features. You cannot enable dev hub in the namespaced org. Nor can you create 1GMP in the dev hub.

The Salesforce documentation clearly states:



- 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.

You will receive the following warning if you try:

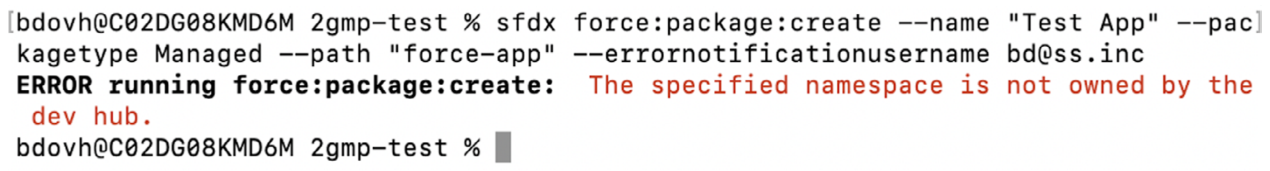


AppExchange For Developers

Change Developer Settings

This org is a Developer Hub. You can only create managed packages in a Developer Edition org that isn't your Dev Hub.

In 2GMP, you must link your namespaced org to your dev hub. If you don't, you'll get an error saying that the dev hub does not own the specified namespace.



```
[bdovh@C02DG08KMD6M 2gmp-test % sfdx force:package:create --name "Test App" --pac]
 kagetype Managed --path "force-app" --errornotificationusername bd@ss.inc
 ERROR running force:package:create:  The specified namespace is not owned by the
  dev hub.
 bdovh@C02DG08KMD6M 2gmp-test %
```

It prevents just anyone from creating packages for a company to which they don't belong. For example, unless someone belongs to the Financial Force company, they couldn't create a package with the namespace fforce.

You must prove that the namespace you are trying to use belongs to you by linking your namespace org to the dev hub.

6. What are the package version limits?

Some developers have dealt with the failure of the SFDX CLI command they are using to create a new package version. A new package will occasionally fail to be created because the system says it exceeds the limit of Package2VersionCreates.

But, what if you urgently need to create a new version for a client anyways?

The solution is to include the --skipvalidation switch in the SFDX CLI command.

This switch skips validation during package version creation. While you can't promote unvalidated package versions, they have a significantly higher limit than the validated version.

The value of Package2VersionCreatesWithoutValidation is 500, while the value of Package2VersionCreates is 6. By including the --skipvalidation switch in the SFDX CLI command, developers can create a new version for a client that avoids the package version creates limit.

7. What's behind these two common causes of errors

Two common errors that arise are:

1. **ERROR running force:package:install: Mismatching versions**

   This occurs when the subscriber org is on the Winter 21 release, while the package version is built with sfdx-project.json specifying version 51.

   Version 51 is the Spring 21 release.

   The story's moral is never to use the preview version until it is available on all your destination orgs.

2. **ERROR running force:package:version:create: No matching source was found within the package root directory**

   This error arises because the specified source directory is empty. There can be numerous causes for this, such as you copying your source to the wrong directory. You will want to go back and ensure that the correct source directory is named.

8. Can you link dev hubs and namespaces?

You should know that not only is it possible to link the same namespace in several dev hubs, but it's also possible to link several namespaces in one dev hub.

9. Is it possible to create packages in deleted or expired dev hubs?

Another question that developers have asked is that once the dev hub trial org is deleted or expires, is it possible to create new package versions?

The Salesforce documentation suggests the answer is no. Once a dev hub is deleted or expires, its packages no longer work.

10. Can a 2GMP Package dependency be defined to depend on a 1GMP or a package owned by a different dev hub?

This is one of the most exciting developer questions, and the answer is yes.

You'll use the command subscriberPackageVersionId to define a dependency, whether it's on the:

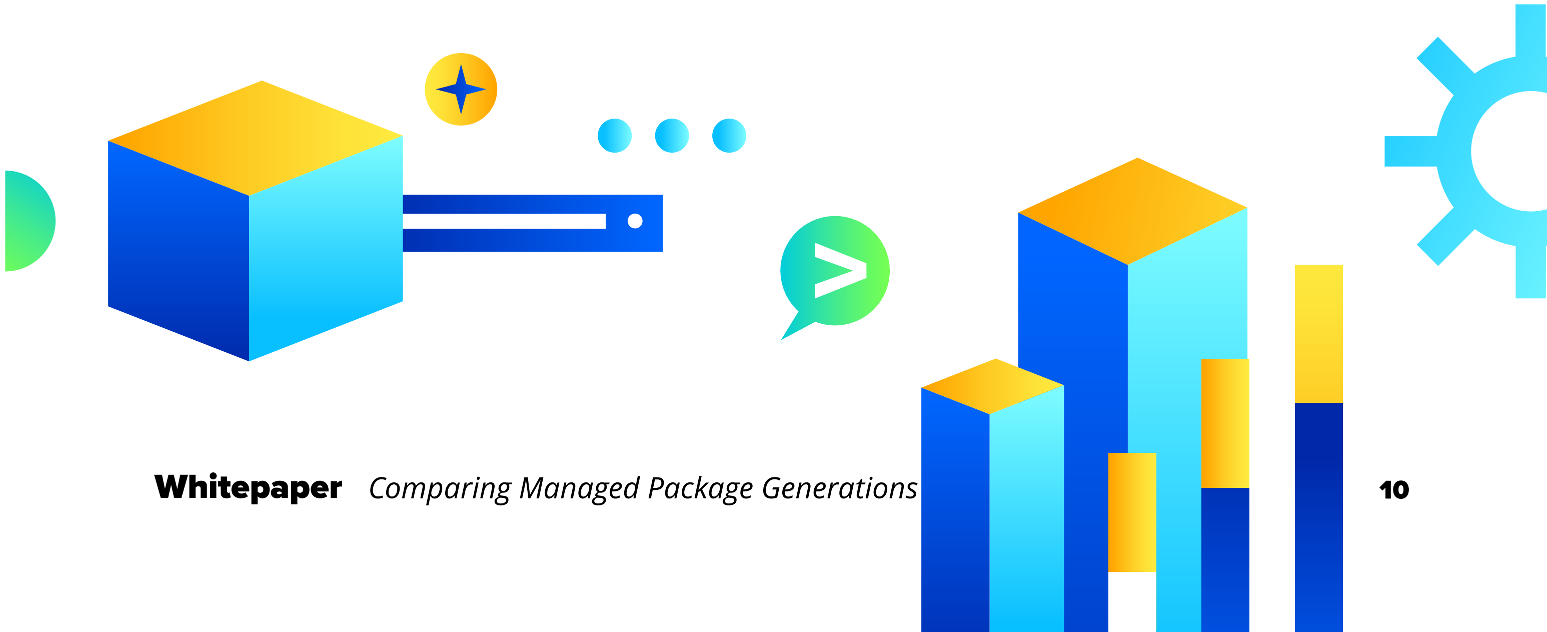
- first-generation managed package
- the second-generation package from another dev hub
- the second-generation unlocked package

CONCLUSION

Salesforce's second-generation managed packages allow for even more partner innovation as they develop, distribute, and manage their apps.

Explaining the differences and similarities between the two managed package generations, their technical nuances, and commonly asked questions highlight the many ways that managed packages can build business success.

LET'S TALK if you're ready to accelerate your developer team's collaboration, enable source-driven development, and reach new heights of business agility with Salesforce.



ABOUT US

SoftServe is a digital authority that advises and provides at the cutting-edge of technology. We reveal, transform, accelerate, and optimize the way enterprises and software companies do business. With expertise across healthcare, retail, energy, financial services, and more, we implement end-to-end solutions to deliver the innovation, quality, and speed that our clients' users expect.

SoftServe delivers open innovation, from generating compelling new ideas, to developing and implementing transformational products and services.

Our work and client experience is built on a foundation of empathetic, human-focused experience design that ensures continuity from concept to release.

We empower enterprises and software companies to (re)identify differentiation, accelerate solution development, and vigorously compete in today's digital economy-no matter where you are in your journey.

Visit our **website**, **blog**, **LinkedIn**, **Facebook**, and **Twitter** pages.

NORTH AMERICAN HQ

201 W 5th Street, Suite 1550
Austin, TX 78701 USA
+1 866 687 3588 (USA)
+1 647 948 7638 (Canada)

EUROPEAN HQ

30 Cannon Street
London EC4M 6XH
United Kingdom
+44 333 006 4341

APAC HQ

6 Raffles Quay
#14-07
Singapore 048580
+65 31 656 887

info@softserveinc.com
www.softserveinc.com

softserve