Spotlight: **Managed & Unmanaged Packages**

If you have been in or around Salesforce apps, you may have heard of packages, particularly managed packages. Aside from being a set of fancy sounding buzzwords, what does a managed package actually do?

Well to start, a 'package' is a bundle of components that make up an application or piece of functionality. Packages are typically created so that their components can be clustered in a container for code migration, editing in an IDE, or distribution to other companies. By default, packages are private, but can be shared via Salesforce or the AppExchange.

Packages come in two forms: managed and unmanaged.

Managed Packages

Managed packages, as we stated before, are a bundle of components that function together as an app or piece of functionality. However, managed packages do not provide the source code of many of the components, and are not intended to be modified significantly by the end user. Managed packages provide intellectual property protection, as well as a way for the package creator to offer upgrades to the package and to provide support for license enforcement features. This is the most common and familiar format for any sort of software app ex. Salesforce, Apple Store Apps, Google Store Apps. In addition, most developers of a managed package will provide maintenance and upgrade services for the managed package.

Unmanaged Packages

Unmanaged packages are a bundle of componentes where the end user does have access to the source code. Basically, the user has a complete copy of the original, which allows the user to continually modify the package as they see fit. While this gives users complete freedom, it also makes unmanaged packages less 'user-friendly', requiring a dedicated and knowledgeable IT team to maintain and upgrade them. After an installer receives their

unmanaged package, generally speaking, the seller/developer will no longer be involved unless specific work is contracted later.

For all intents and purposes you can think of unmanaged packages as creating a hard product and handing it off to someone. If the creator deletes or modifies the package later on, the previous versions are unaffected. Because of this, unmanaged packages are primarily used for one-time distributions and are ideal for code sharing or migrating application templates.

Even though the user the of the unmanaged package has direct access to and visibility of the code, they cannot sell or relicense the code, no matter how many changes they have made, without the permission of the original creator.

Key Differences

"Ok" you say, "But what does all of this mean for me, practically speaking?" Well, we can break it down into three primary areas of difference between a managed package and an unmanaged package: **price, maintenance and control.**

Price & Maintenance

An unmanaged package is generally priced higher than a managed package, for the exact same functionality. Why? The user of the unmanaged package is essentially paying a premium for the ability to see and access the code.

Because maintenance, growth and bugs are inevitable, it is highly recommended the users of unmanaged packages have a large and knowledgeable in-house IT department to maintain the package.

In a managed package however, all future development costs and maintenance are shouldered by the seller. If the end users would like to enjoy the fruits of that development, they will most likely need to pay for upgrades or extra features.

Control

An unmanaged package undoubtedly gives the end user more control over package changes and troubleshooting. However, in worst cases, it can also lead to a tough to manage, and easy to break system. It all depends on how good the user's IT team is, and if they are able to properly manage the package. It should be said however, that sometimes when it comes to development and coding, the best people to make any modifications are the ones who built it.

To that end, an unmanaged package is more difficult for the original seller/developer to maintain and upgrade. As the user of the unmanaged package makes changes to the

system, those changes are not visible to the seller and therefore the seller will have difficulty maintaining and pushing upgrades to the system as some of the changes may get in the way of those upgrades.

On the other hand, with a managed package, the developer controls everything when it comes to what upgrades get made, how they are function and how quickly it gets done.

The “Easy on the Eye” Breakdown

Unmanaged Packages	Managed Packages
More expensive up front	Less expensive up front
No new versions available	New version upgrades for a fee
Requires significant end-user IT development and maintenance	Requires minimal end-user IT work
Gives user access to the source code	No user access to the source code
No continuing developer support	Continuing developer support
End-users can better control changes and troubleshooting	End-users have few options for making changes
Most common for open source projects	Most common for complete apps and software
Challenging for original developer to fix bugs and make changes	Easy for original developer to fix bugs and make changes

The Bottom Line

So what does it all boil down to? It really depends what each user needs and wants out of the application, and what their abilities to maintain and develop that package are. Before you decide on what type of package is right for you, ask yourself some of these questions:

- Do I have a dedicated IT team?
- How much to I want to spend on and deal with maintenance?
- How important is it that I have source control?
- What are my end goals from this package?

Whatever your answers, hopefully this guide has helped you get a better understanding of just what managed and unmanaged packages are.

Want More?

Take a look at this helpful article from Salesforce:
https://help.salesforce.com/articleView?id=sharing_apps.htm&type=0

Contact us to for more info on our managed packages for Salesforce:
http://go.cloudmybiz.com/getmoreinfo