

Application Packaging Process - The End-to-End Tutorial

For the past 10+ years, I have worked with various projects and types of organizations as an Application Packager. During that time, I have witnessed some "elementary" mistakes that could have been easily avoided by having a clear understanding of the end-to-end application packaging process.

To help you succeed at becoming an efficient application packager, I will be sharing my experience on what has worked for my customers and some tips that could make your application packaging process more seamless.

Although this article is mostly intended for application packagers, its content is also valuable for people in executive positions who need to have an overview of the real-life activity of their application packager teammates.

If you're searching for an application packaging tool, I recommend checking out this page: MSI, MSIX, and App-V Packaging Software to see how it may help you create application packages.

"I do not fix problems. I fix my thinking. Then problems fix themselves." - Louise Hay

In this article:

- Application Packaging Industry Trends
- What is Application Packaging and what's its role?
- High-level objectives and benefits
- A Practical End-to-End Application Packaging Process along with some recommendations on how to overcome the challenges and mitigate risks.



Application Packaging industry trend - What skills are companies looking for?

Nowadays, when a company is looking to hire an Application Packager, they expect for that person alone to take care of the entire Application Packaging process. They have this expectation for both permanent employees and contract-based packagers.

There are two main reasons why companies take this approach:

- to cut down costs,
- to improve app packaging process efficiency by speeding up the process and reducing the time an application takes to be packaged and deployed to production devices – the more people are involved in the process, the slower the process is.

The downside of this approach is the difficulty to find the candidates with the right skills, and who fully understand the end-to-end packaging process.

What is Application Packaging and why do we need it?

Modern organizations commonly have hundreds to thousands of software products installed on client computers. Each software product comes with its own unique installation and configuration requirements, making the management of the software products a more complex process.

And it will become even more complicated with Windows 10 and Windows as a Service model. I will go into why later in this article.

Application Packaging helps organizations in need of a way to take the burden off their IT support teams while providing an improved end-user experience.

If we look over the Application Packaging benefits, we can find that it:

- ensures a consistent, stable and reliable standard environment;
- increases the efficiency of software management by streamlining the software deployment along with any customization needed;
- mitigates security issues;
- decreases risks for business disruption;
- helps reduce the on-going administration and support costs.

We will go through each and every step of this process, but before, let's clarify an important aspect, Application Packaging is not a miracle recipe for all problems.

Does Application Packaging solve all your problems?

There are many companies that expect to resolve all their application related issues, including compatibility issues, just by packaging each and every one of them.

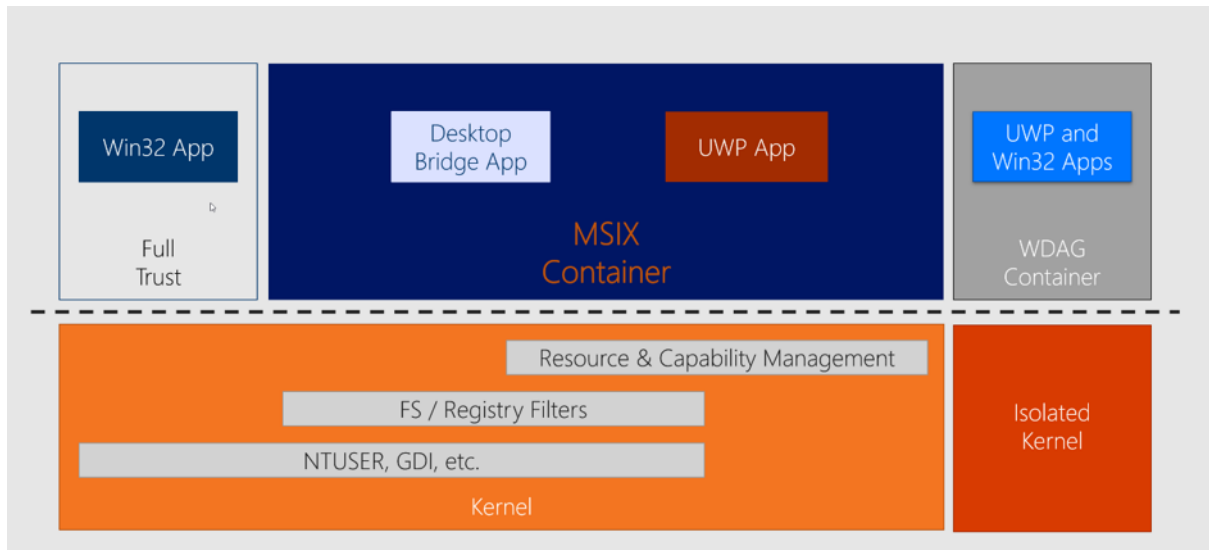
Although there are some compatibility issues that could be sorted by remediating the application and including the fixes within the package, this is not possible for the majority of them.

Solving application compatibility issues in packaging relies on containerized packaging solutions.

Some well-known containerized packaging solutions include:

- App-V, which enables applications to run in their own container on the client computer;

- The new MSIX packaging format, which is based on the same concept of containerization as App-V.



Also, Microsoft provides some basic, but useful, resources on how to tweak an older application and make it work with Windows 10.

We can't forget about Application Compatibility Toolkit (ACT) - the tool developed by Microsoft to help enterprises fix application compatibility issues that may occur due to the changes between Windows Operating System versions. This tool came about 2010 when enterprises were looking to upgrade from Windows XP or Windows Vista to Windows 7. Since then, once Windows 10 was released, the majority of ACT functionalities were moved to Windows Analytics - which was recently replaced on 31st January 2020 by Desktop Analytics.

End-to-end Packaging Process

Now that we understand how that end-to-end packaging is helpful but is not necessarily the answer to all of our problems, let's go through the end-to-end packaging process. Now, in most organizations, the end-to-end packaging process consists of 3 main steps:

- 1. Application Discovery
- 2. Application Packaging
- 3. UAT (User Acceptance Testing)

Organizations that care about the quality of their packages have also introduced an extra step: the review of the packages created by a senior packager (QA).

From a high-level perspective, the process must look like the one in the diagram below.



Application Discovery

The first step is the Application Discovery, and it consists of the following:

- validating the application source file
- ensuring that the application is fully functional within the organization environment and that it works as expected.

Through this step, all the requirements and details of the application are collected and recorded accordingly. Keep in mind that some applications may require less discovery than others. Also, sometimes vendor support is needed if the application fails to install or work as expected due to any compatibility issue or misconfiguration within the organization environment.

This is not the time to be superficial since the future package is going to be created based on the requirements and the application details recorded during the discovery.

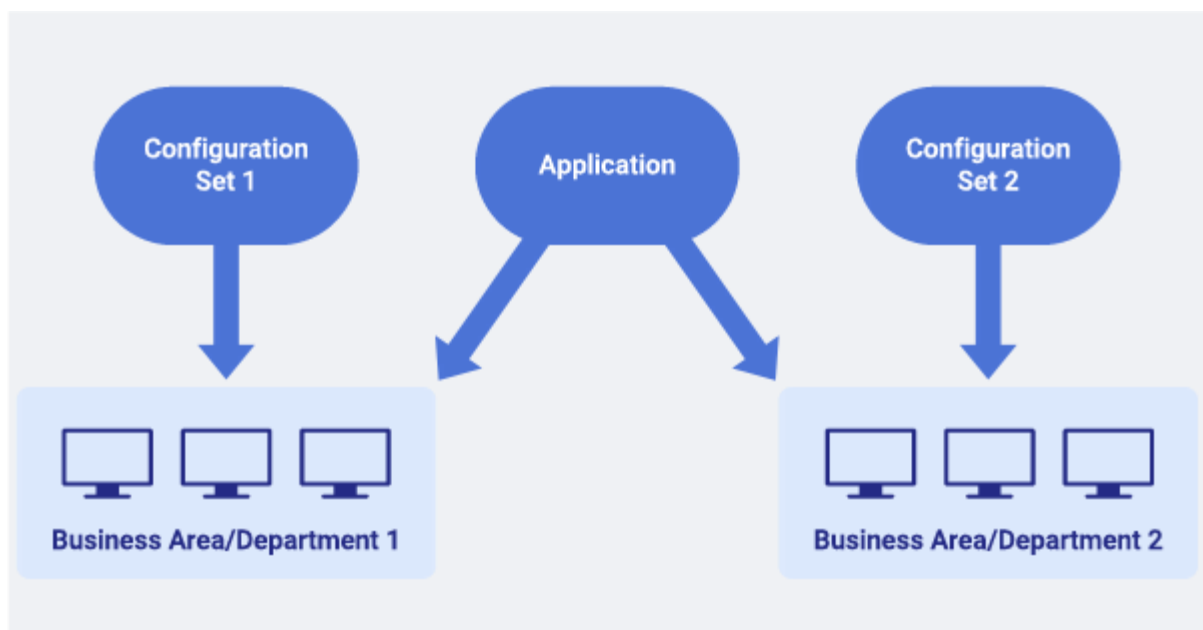
You will realize that it is a waste of time to package an application only to find out later (when the packaging is completed, or worse, after the package is released to production) that the application is not fit for the purpose it was created or that the package has missing configurations within it.

RECOMMENDATION: Plan the application discovery carefully and make sure to get the correct outputs, which will be then used for packaging your application.

During the discovery phase, also gather the following information that could significantly affect the packaging process :

Is the application used differently by users from different business areas? Does each business area require different sorts of configurations?

Depending on the responses to the previous questions, it may be worth having separate packages: one package for the application and separate packages for each business area configuration. Then, each user will get the corresponding configuration package along with the application package.



Along with MSIX, Microsoft also released MSIX Modification Package - the packaging format meant to store the customizations of the application.

Along with MSIX, Microsoft also released MSIX Modification Package - the packaging format meant to store the customizations of the application.

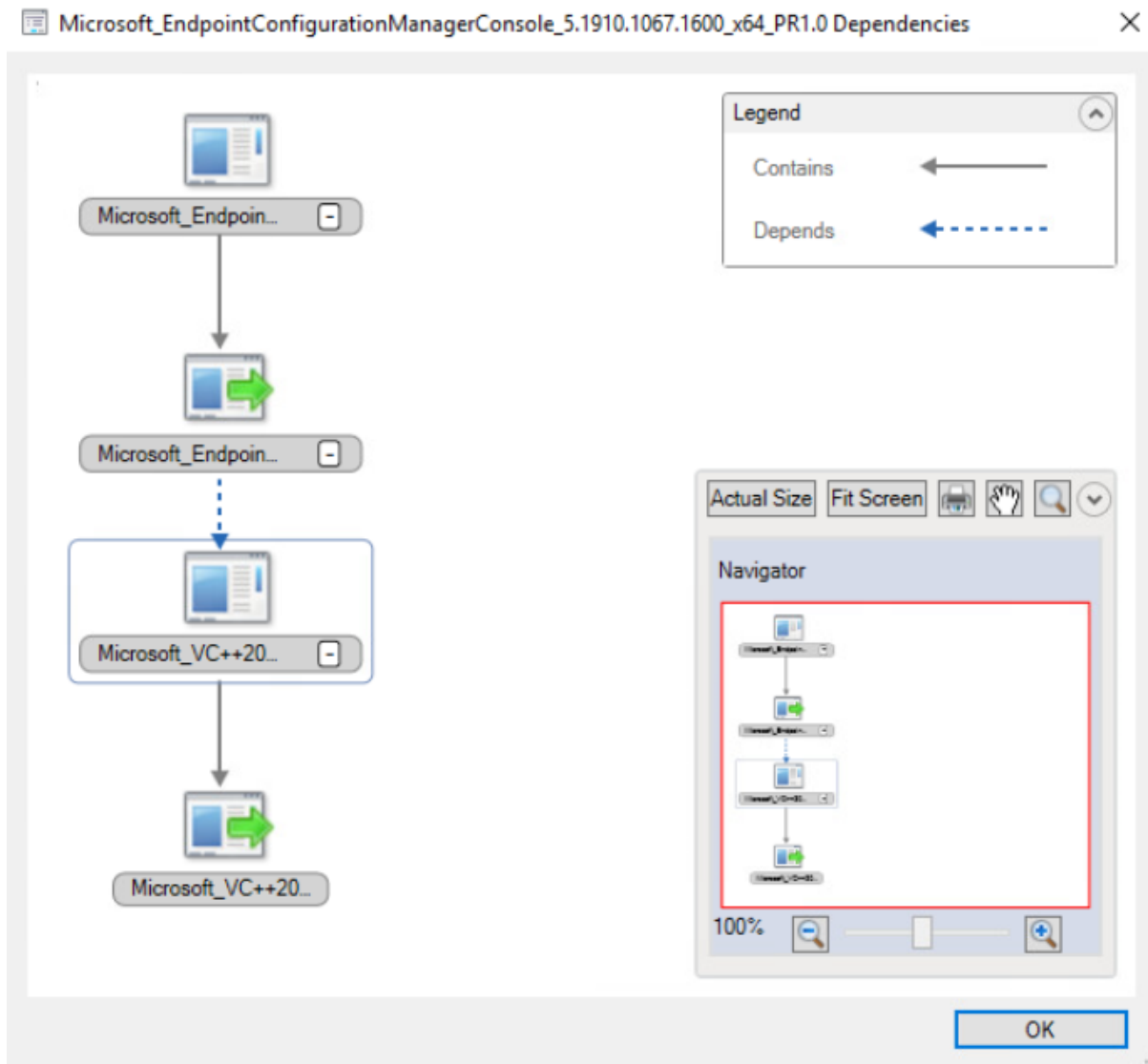
Does your application have any prerequisites?

If the answer is yes, you might want to take them out of your package and create separate packages for them. The reason is so that you will be able to reuse them in case other applications need the same prerequisites.

This can happen in a new version release of the application or when you need to rework the package for some reason. If you've already taken these measures, it will be much easier to package just the application rather than to repackage the whole suite.

At the same time, this helps you to keep your devices up-to-date and mitigate any security vulnerabilities in case a new patch is released for the prerequisite.

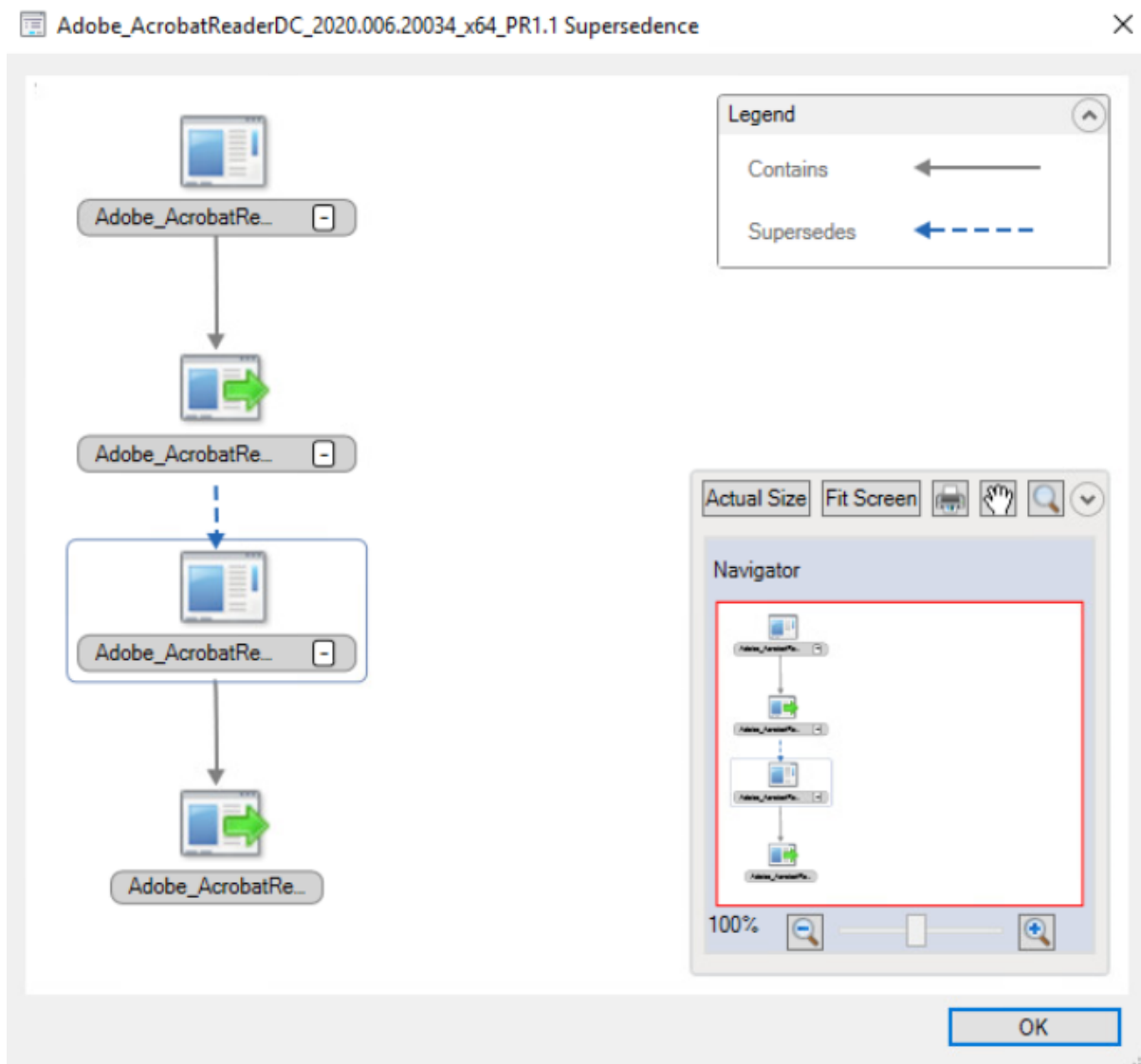
However, this is arguable, especially if App-V is your preferred packaging delivery format. And sometimes, it can be more challenging to keep them as separate packages rather than sequencing the whole suite as a single App-V package to run in its own container.



Is your package meant to replace any other package?

Managing and maintaining a high number of packages could be difficult regardless of the deployment tool you use. So, it is a good practice to keep this number down.

Before the application is packaged and imported into the Configuration Manager or Intune, it is important to determine whether it's a new package revision of an existing packaged application, a new version, or a completely different application. The same consideration applies to dependencies - both the Configuration Manager and Intune come with their supersedence mechanism to handle upgrades, which is quite robust and seamless to use.



Application Packaging

Application Packaging covers the actual package creation based on the requirement and details gathered in the Discovery stage. There is not much left to be said here - depending on which packaging format you prefer to use within your organization, there are all sorts of best practices articles online

Check out our tutorial, if you want to find out more about Repackaging Best Practices.

To see a practical example on how you can create an application package with a free Windows installer tool, check out this video tutorial:

User Acceptance Testing

Application UAT is the step of the application packaging process before the packaged application is deployed to production and it consists of validating the package created and making sure that the packaged application is fully functional and works as expected – basically, it must behave the same as the vendor source file tested earlier in the Discovery stage.

Application Deployment

Application Deployment is the process of installing a package created using a software management tool such as Configuration Manager (formerly known as SCCM) or Intune.

With Configuration Manager and Intune, you can not only install your application packages but also uninstall them and generate various queries and reports. Additionally, depending on your requirements, you can target both device and user collections.

Best Practices for User Acceptance Testing, Deployment and post Deployment

1. Use Virtual Machines for UAT.

The majority of companies use tools like VMWare or Hyper-V to host their virtual test machines used to perform the UAT, mainly because these tools come with the great capability to revert the virtual machines to a previous state within seconds.

This comes in handy considering that each application must be tested on a vanilla device with no other applications installed on it, apart from the ones included in the build.

Those virtual machines must be a replica of the actual production devices (the same applications installed on them, the same GPO applied to them, and the same settings, etc.) so if there is an issue with the package it can be picked up in UAT rather than after the package goes live.

2. Full comprehensive UAT.

Some companies may have defined test cases for each application within their environment. Although this is something nice to have, it is not mandatory. A full comprehensive test will do as well.

If the packaged application is used by multiple business areas in different ways - the tests should include each of those business areas' scenarios to make sure there is no issue with the package created in terms of its functionality.

3. Import it into Configuration Manager or Intune first, then get it UAT'ed.

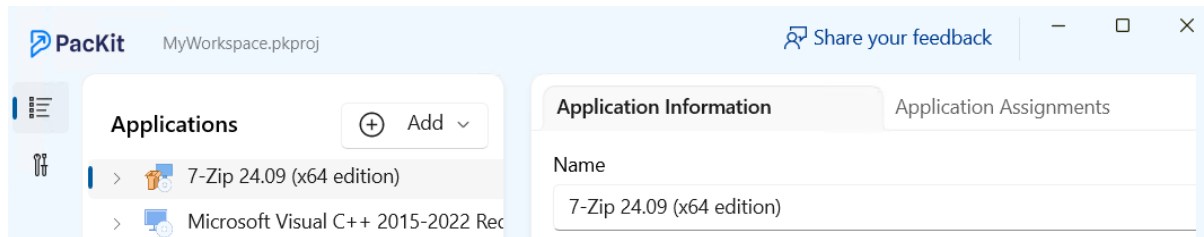
Some companies conduct UAT before importing the application into the Configuration Manager or Intune. These companies usually use utility tools – PsExec being one of the most well-known, to mimic the installation via Configuration Manager or Intune. This is not necessarily wrong. However, there are very few scenarios where a package may behave differently once imported and deployed through the Configuration Manager or Intune. To minimize potential issues, it is recommended to perform the UAT only after the package has been imported and set into the Configuration Manager/Intune.

Before adding a Win32 app to Microsoft Intune, you must convert the application package into the .intunewin format by using Microsoft-Win32-Content-Prep-Tool

You can either do that manually or use a tool like PacKit which reduces the manual effort and streamlines application management by enhancing the post-packaging process. With capabilities for importing packages into MECM and Intune and then deploying them, PacKit empowers IT professionals to efficiently organize, manage, and deploy applications in a much more effective way.

It offers features such as integration with the PowerShell App Deployment Toolkit (PSADT) for automating script generation and automated .intunewin file generation for Intune deployments.

Moreover, it integrates with WinGet, allowing you to easily import preconfigured packages directly from the WinGet repositories into the PackIt workspace.



Check out the PackIt features page, for more details about it.

Once UAT is signed off, the packaged application can be deployed to production devices for those users who need it. And, you might think that from here onwards it will be a smooth and straightforward process. Unfortunately, it is not always the case.


4. Phased and controlled Deployment.

Although the package was tested and UAT was signed off, there is always a risk for something to go wrong when deploying it out to production devices.

Companies should not take any risk. If they take it - it should be mitigated as much as possible, so a regression plan must be in place in case something goes wrong. Therefore, from a deployment perspective, a phased and controlled deployment approach is something to take into consideration. This can be achieved by manually adding members to the collections, or using the Phased Deployment feature included in the Configuration Manager.

Create Phased Deployment

×



General

General

Settings

Phases

Summary

Progress

Completion

Configure general settings for this phased deployment

Phased deployments automate a coordinated, sequenced rollout of software for managing multiple deployments.

[Learn more about phased deployments](#)

Name:

Microsoft_VC++2015-2019_14.24.28127.4_x64_PR1.0

Description:

☒ Automatically create a default two phase deployment

First Collection:

Phase1-Pilot_Microsoft_VC++2015-2019_14.24.28127.4_x64_


Browse...

Second Collection:

Phase2-FullRollOut_Microsoft_VC++2015-2019_14.24.28127.4_x64_

Browse...

☐ Manually configure all phases

 Distribute the content for the application to a distribution point before creating the phased deployment.

< Previous

Next >

Summary

Cancel

5. Rework the existing package to include the fix if it fails.

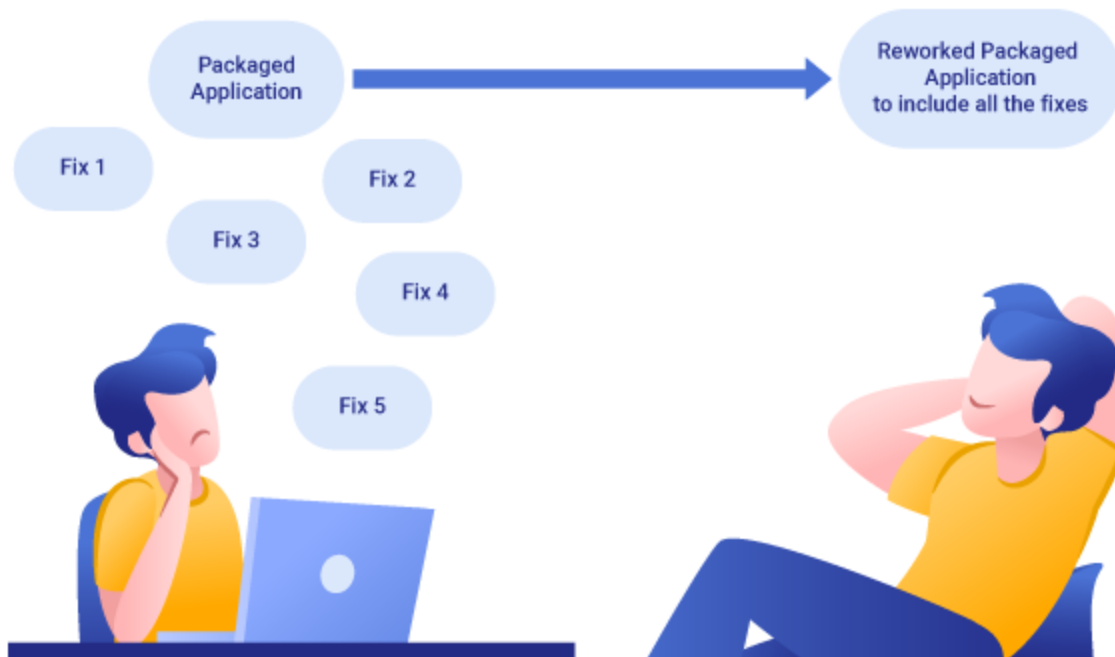
However, a production device has lots of applications installed on it, so conflicts may appear between applications. This sort of issue must be investigated - to find the root cause of the issue and to eventually rework the package to include the fix.

The app conflict issues are not applicable to containerized packaging formats such as AppV or MSIX where each application runs within their own containers.

What if your packaged application went live and got installed on production devices carrying an issue? Fight the urge to get the fix packaged in a separate package and deploy that along with the existing packaged application.

Here's the way to solve this:

- Get the package reworked to include the fix
- Deploy the reworked package.



In the short-term, it could be a lot quicker to have a separate package just for the fix, but in the long-term it could cause a lot of headaches, especially if they are not managed properly from a deployment perspective. And, one package is definitely easier to manage than two.

6. Keep your applications up to date.

Starting with Windows 10, Microsoft introduced a new model to build, deploy, and service Windows, known as Windows as a Service (WaaS) .

With WaaS, you get incremental updates (twice a year for Windows Feature Updates and every single month for Windows Quality updates) instead of getting a new version of Windows every three years.

According to the Microsoft Windows lifecycle fact sheet, Windows 10 Enterprise Feature Updates released in March are serviced for 18 months from the release date; whereas the one released in September is serviced for 30 months.

Basically, this is the time in which each company will have to test and eventually fix their applications if needed. And it is much more likely for an out-of-date application to break with the release of a new Windows Feature Update.



7. Application model VS Package model in Configuration Manager.

Whilst Microsoft continues to support the use of the Package model in all their Configuration Manager Current Branch versions, I personally found the Application model much more powerful and more flexible than the Package model, and therefore I recommend to use the Application Model when you want to deploy a proper application.

The package model may be more suitable than the Application model when you want to deploy a tool or a script.

But for proper applications, you must use the Application model as it is the only one that comes with features like dependency, supersedence, detection methods, application groups, and others

When using the Package model, you cannot deploy an App-V package natively.

8. Set supersedence in the Configuration Manager or Intune where necessary.

First, you must decide whether you want to upgrade or replace an application.

Of course, there may be reasons for you to deliberately want to keep two or more different versions of the same application in production. This is usually when:

- you have a limited budget and you choose not to upgrade all users of an application to the latest version available, so you still want to keep the package for the old version for the time being;
- the vendor deprecated some functionalities in the latest version-specific functionalities that are still needed by some of your users to do their work.

However, even if you face any of these challenges and need to keep two or more versions of the same application in production, you still should set the supersedence between them when you import them into the Configuration Manager or Intune.

See Configuration Manager app supersedence, and Intune Win32 app supersedence for more details on what to take into consideration and how to set supersedence in the Configuration Manager Current Branch.

9. Periodic housekeeping.

New packages are created and imported into the Configuration Manager or Intune all the time. They have to be reviewed periodically; the main aim here is to get rid of the ones that are not needed anymore and keep the number of packaged applications as low as possible.

10. Application Rationalization

Periodic housekeeping is the bare minimum which each organization must do. However, since over 75% of IT budget is allocated to operating and managing applications, some organizations might want to go even further and put an application rationalization process in place in order to lower this percent and reduce the costs.

Application rationalization is the process of assessing all business applications within an organization in order to identify which ones should be retained going forward, replaced, retired or consolidated. Application rationalization in its true sense of the word will require an experienced team of SMEs

with the right mix of skills and knowledge. This usually involves additional resources and requires additional budget, but long term it will lead to cost savings and business benefits.

In order to take the full benefit, Application Rationalization must be an ongoing process with regular assessments of each software product used within the organization.

11. Retire, then uninstall, then remove.






If a packaged application is not needed anymore, don't just delete it from the Configuration Manager or Intune. In the first instance, you must take care of the existing devices where the application is installed. Otherwise, the application will carry on being installed on all the devices where it has been deployed to and you will not have an automatic mechanism to remove it from there.

So, you must first retire it to ensure that specific packaged applications will not be deployed to any new devices. Then, you must uninstall it from all the devices where it has been deployed via the Configuration Manager or Intune. And after those steps have been completed, you can safely remove it from the Configuration Manager or Intune.

12. Nice and tidy Configuration Manager or Intune environment.

Let's not forget that Application Packaging as a service is mainly addressed to medium and large organizations with thousands of users who are very likely to use hundreds of applications.

Therefore, having a set of standards put in place is mandatory. The naming convention is important and must be one of the first things that must be addressed. Even if it has no direct impact on the functionality of the packages created, it helps a lot if you have a nice and tidy repository and the Configuration Manager or Intune environment for your packaged applications.

Icon	Name	Deployment Types	Deployments	Status	Manufacturer	Superseded
	Adobe_AcrobatReaderDC_2020.006.20034_x64_PR1.0	1	0	Retired	Adobe	Yes
	Adobe_AcrobatReaderDC_2020.006.20034_x64_PR1.1	1	1	Active	Adobe	No
	Microsoft_EndpointConfigurationManagerConsole_5.1910.1067.1600_x64_PR1.0	1	1	Active	Microsoft	No
	Microsoft_VC++2013_12.0.40664.0_x86_PR1.0	1	1	Active	Microsoft	No
	Microsoft_VC++2015-2019_14.24.28127.4_x64_PR1.0	1	1	Active	Microsoft	No

13. Comprehensive Discovery and UAT sign off documents.

It must look more like paperwork, but it is good to have all the details recorded into the relevant packaging documents (discovery and UAT). They are very useful if you have to rework a package, especially if that package was created by another member of the team who left the company in the meantime.

14. Avoid manual installs.

Sometimes, for those applications which need to be installed on a small number of devices, you could say that it is not worth to package and deploy them via the Configuration Manager.

You could just manually install them. Well, that is a good rationale, as you could ensure that the number of devices will stay the same. But, could you? That is almost impossible to achieve within medium or large organizations.

The same application that you manually installed might be later needed for another department. Or a new joiner of a team may need it.

And therefore, the number of devices where the application is needed could grow considerably, as well as, the effort put to manually install the application on them.

Regardless, you must first test it on a test device, preferably a Virtual Machine, and ensure the application works fine and it does not break other applications, especially the core ones.

For more details, you could check out our [Why Your Manually Installed Applications Count Should Be Zero](#) article.

[Video Tutorial - The Application Packaging process explained](#)

Conclusion

The main goal of Application packaging is to reduce the administrative and support cost for the companies while standardizing their environment. This can't be achieved just by having good quality packaged applications that follow application packaging best practices.

Each application is different; therefore each application must be reviewed and assessed individually. On top of that, each packager must have a clear picture of the end-to-end packaging process, starting from the moment the packaging request is raised to when the packaged application is not needed anymore and it must be decommissioned.

I gathered all the above points as basic recommendations that can be applied in most organizations.

With Windows 7 now being out of extended support, organizations must upgrade to Windows 10 as soon as possible if they have not done it already.

Make use of this opportunity to assess your applications, review the technologies your organization uses and update your standards and processes to accommodate the new changes, including the lifecycle of the application. This will help you to mitigate the risks and simplify the deployments.

All of this to achieve a single goal: to reduce the overall costs.