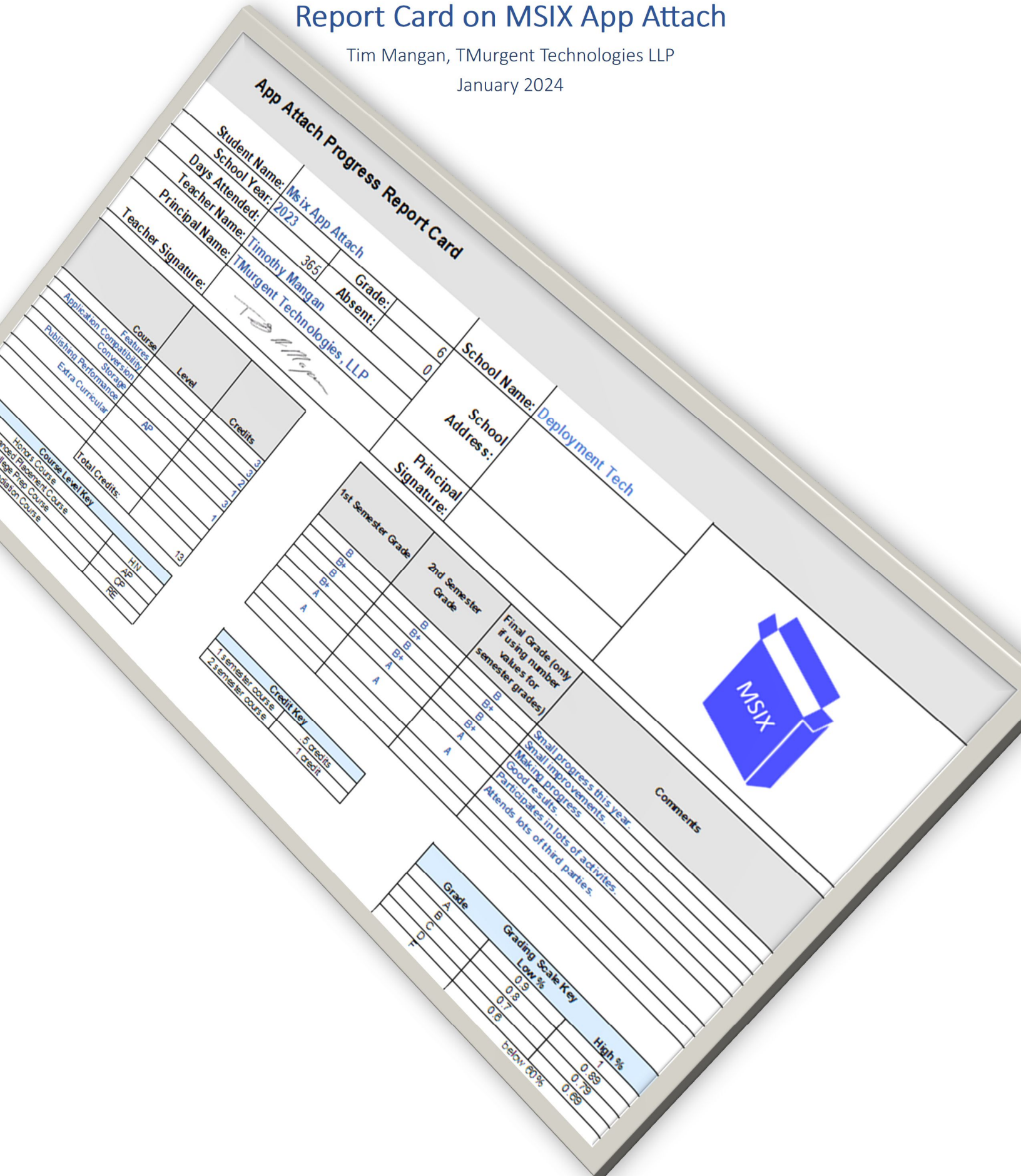
Report Card on MSIX App Attach

Tim Mangan, TMurgent Technologies LLP

January 2024



Introduction

App Attach is a technique that uses “application layering” methods to quickly mount and publish MSIX applications into a non-persistent (pooled) VDI scenario and scenarios involving shared operating systems like Remote Desktop Services and Multi-User Windows 10/11.

MSIX App Attach is the Microsoft-branded feature for use with Azure Virtual Desktops that uses the App Attach technique for managed deployment of applications to AVD desktop sessions. Microsoft appears to be renaming the “Msix App Attach” feature name as “App Attach in Azure” in 2024.

Other vendors with application management features also use the App Attach technique, including Citrix, VMWare, and AppVentiX. They may call it MSIX App Attach or just App Attach.

This paper will look at features of the technology, any issues with application compatibility, and performance of the underlying App Attach technology used by all vendors.

Features

App Attach uses a different format for the package definition from the “normal” MSIX format. Actually, there are multiple choices of new formats to choose from. These formats are better suited to the needs of the application layering technique to ensure fast availability of the applications assigned to a user after they log onto a Windows session.

In general, this means that the package format is a windows disk partition (technically volume) that is remotely mounted, rather than copied into the user’s VM, and integrated into the user environment. This is done in a three-step process of Mounting, Staging, and Registering.

In single-user OS, these are all done for each package. In a multi-user OS, the mounting and staging steps may be skipped for packages already added to another user.

Once the application is registered, runs inside the same MSIX container that would have been used if the MSIX format was used to deploy the package. This means that the application should behave as if the application was natively installed (subject to general MSIX limitations).

App Attach adds no new capabilities to standard MSIX deployment and execution other than the speed of getting the package ready for the user. As there is no free lunch, this means that when the application is running, the package files will be accessed from over the network via this remote mounted share. Given that in practice, the typical scenarios involve the main OS disk also being a virtual disk accessed over the network, this is of little concern and actually a benefit in the reduction of write IOPS to the main virtual disk.

Application compatibility

We have two things we need to talk about here. Application Compatibility with MSIX, and additional compatibility concerns with App Attach.

From time to time, we hear from customers that come to us with lists they discovered on the internet of things that don't work with MSIX. Most of these issues are outdated given the amount of work spent on improving the app-compat story for MSIX over the last 5 years.

App Attach originally imposed additional restrictions, some of which have been addressed too.

Base MSIX app-compat issues:

While not a complete list, here are the things we hear about, *and where the issues stand today*:

1. MSIX package format does not support Device Drivers. Applications which contain Device Drivers may not work as expected when converted to MSIX package format. It is recommended not to convert such applications to MSIX.

   This is mostly still the case, which is consistent for most app virtualization and layering products. The normal approach for all of these deployment methods is to separate out the driver from the application and treat it as a dependency.

   Under MSIX, it is possible to list this dependency in the AppManifest file of the package and it would be automatically installed, if available. While Microsoft's MSIX Packaging Tool uses this technique successfully, this isn't quite practical for most applications with drivers so we just don't do that and treat the drivers as a separate requirement, either in the base image or delivered via static deployment tools like ConfigMan, Intune, or even AutoPilot.

2. MSIX package format does not support Windows/NT Services. Applications which contain Windows/NT Services may not work as expected when converted to MSIX package format. It is recommended not to convert such applications to MSIX.

   This is no longer true. Initially services were not supported under MSIX. Support is available in current generally supported OS versions. When support for services were added to MSIX, initially App Attach did not support the services, but this was taken care of long ago and is no longer an issue.

3. Shortcuts are the entry point for the MSIX packages. Applications with no shortcuts are not recommended to be converted to MSIX.

   This is wrong. There are many forms of entry-points other than shortcuts. Additionally, even "middleware" that exposes no entry-points other than being able to be called by the exe name by another application (think javaweb.exe) can be successfully packaged. This might be done as a Modification Package, a Dependency Package, using Shared Package Container, or simply including the middware component in the application package that needs it. SPC only works on Windows 11 22H2 or above.

4. Applications having conditionalized components are not recommended to be converted to MSIX.

This is tricky, and mostly misleading. The statement can be made about any form of application packaging that includes a recapture operation, including MSI repackaging. Using standard best practices in packaging avoids these issues.

5. MSIX package format does not support Unsupported .Net Framework Version. Applications having unsupported .net framework version below 4.6.2 are not recommended to be converted to MSIX.

   I guess this is true, but so what? That is because no support is available for an application deployed using any method (including MSI) that is using .Net Framework 4.6.2 and below because those frameworks themselves have fallen out of support. But 4.8.1 is supported and those applications should work fine when deployed on a system with 4.8.1.

6. MSIX package format does not support elevated privileges. Applications having shortcut exe which require elevated privileges are not recommended to be converted to MSIX.

   This is no longer true (I think it was fixed in the 1704 OS release). Exhibits A and B for this are my tools PsfTooling and TMEditX, both of which are MSIX packages that elevate. I'll note that there is a trick to getting an entrypoint application that does not elevate to start a child process that requires elevation, but it can be easily done.

Based on what I see in the field, the following issues are valid today, even if not on that list:

- Applications requiring certain types of Shell Extensions, such as a DragAndDrop handler.
- Applications with lots of COM components that need to be used by more than one exe in the package. Most COM based apps don't have this issue, but there are a number that do.
- Registry Deletion Markers. We need this for handing apps that need an older version of Java.
- Plugins to Office. Add to this any MSIX package that needs to work with a component that will run in a different containerized solution such as App-V or Click-To-Run.
- It doesn't work and we don't know why. We started out 5 years ago with 75% of the apps in this bucket. We are down to about 15%.

The first three items on the list are now considered to be items possibly fixable by the PSF. These are being worked on. Microsoft is believed to be working on Office.

App Attach specific app-compat issues:

While Microsoft has previously addressed the issue with Windows Services under App Attach deployments, the only outstanding issues I am aware of would be:

- Packages that indicate Dependency packages. While the normal installation of an MSIX package would automatically deploy the dependency (if not already present) when the package is installed, the MSIX App Attach deployment for AVD does not currently do this automatically. You can, however just assign the dependency in the Azure console. Even though Dependency Packages have been available from day one for MSIX, we are only starting to make use of them, mostly for well-known Framework dependencies like VCRuntimes and WebView2.
- Shared Package Container. The Azure console does not have support for this, but in a bind it could be pre-defined as part of the image as the packages to not need to be present to add the sharing rule.

Getting from MSIX to App Attach format

Although there is nothing preventing a software vendor from making their software directly available to you in an App Attach format, and we are seeing a small increase in vendors releasing in an MSIX format, we are unaware of any releasing in the App Attach formats today. So whether you get the MSIX package from the vendor, or create your own by repackaging, you'll still need to convert it.

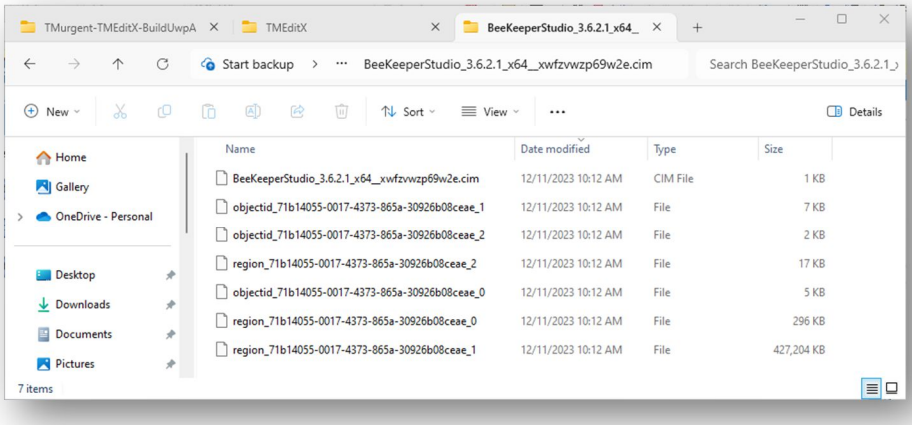
There are three different format options available to you, and not all tooling that is out there supports all of them.

Format	Description	Limitations
VHD	Uses the standard VHD format used by most app layering solutions, so lots of tooling is available.	255 character file path limit
VHDX	Ditto, but has a larger maximum size. But you probably don't have app packages more than 2TB in size.	255 character file path limit
CIM	Uses new Cim file system for improved performance.	Not much tooling available

In all three of the formats, the MSIX files are stored uncompressed in the image, unlike the original MSIX package where they are compressed. So, on average these images will take up 2.5 times the storage that the MSIX packages do.

The differences between the VHD and VHDX choices are unimportant to App Attach. VHD already supports a 2TB size which should be much larger than any application image you are likely to ever create. Other differences in the format are clearly not going to affect App Attach. The VHDX version of a package will be slightly larger than VHD, but not significantly so.

The CIM image format is, unlike VHD or VHDX, not a single file, but a folder with a handful of files that make up the image. Although you could put multiple of these in the same folder for expediency (each has a unique partial name), the best practice is to keep each CIM separate by storing the files in a folder with the CIM name. In the TMEditX conversion, when selecting the location for the CIM image, you will want to create the folder when picking the location to store the image. The following image shows an example of the files produced that together form this image:



While the CIM format may be harder to deal with, being new and with limited tooling, it is not subject to the 256 character path limit, which affects applications like those including a Python distribution. When working with deploying packages using CIM, the techniques to deploy are different, although it is likely that the tooling you use will hide those differences. More important are the differences in how you debug as even visibility that there is a mounted package changes and you must use different (and more obscure) commands for that visibility. This is a solvable problem at the OS level (by making Cim filesystem a first class citizen), should Microsoft be compelled to do so.

Some of the known tooling for conversion from MSIX to App Attach formats include the following:

Vendor/Tool	Description
Microsoft Packaging Tool	Free. Available in preview build only. Supports conversion output to the VHD format only.
MSIX Hero	Free. Supports conversion output to CIM and PS scripts for testing. May no longer be up-to-date?
TMEditX	Licensed. Supports all three formats. Supports conversion and testing.
AppCure	Licensed.

Storage Requirements

App Attach application images must be stored in a location with relative low latency to the machine or VM where they will be used. This implies that both the storage location and VM are located in the same data center (or Azure region).

Both traditional Windows file shares and methods like Azure Files may be used for this storage.

While MSIX packages are compressed, conversion to these new formats are uncompressed. There is a possibility that some of the files might be compressed in the CIM format, but without proper publicly available documentation on this new format we are not sure. Both VHD and VHDX also have a minimum disk image size of 100MB, so small MSIX packages converted to these formats will require much more storage than the original MSIX files.

The following average sizes were determined from a set of 90 packages we tested:

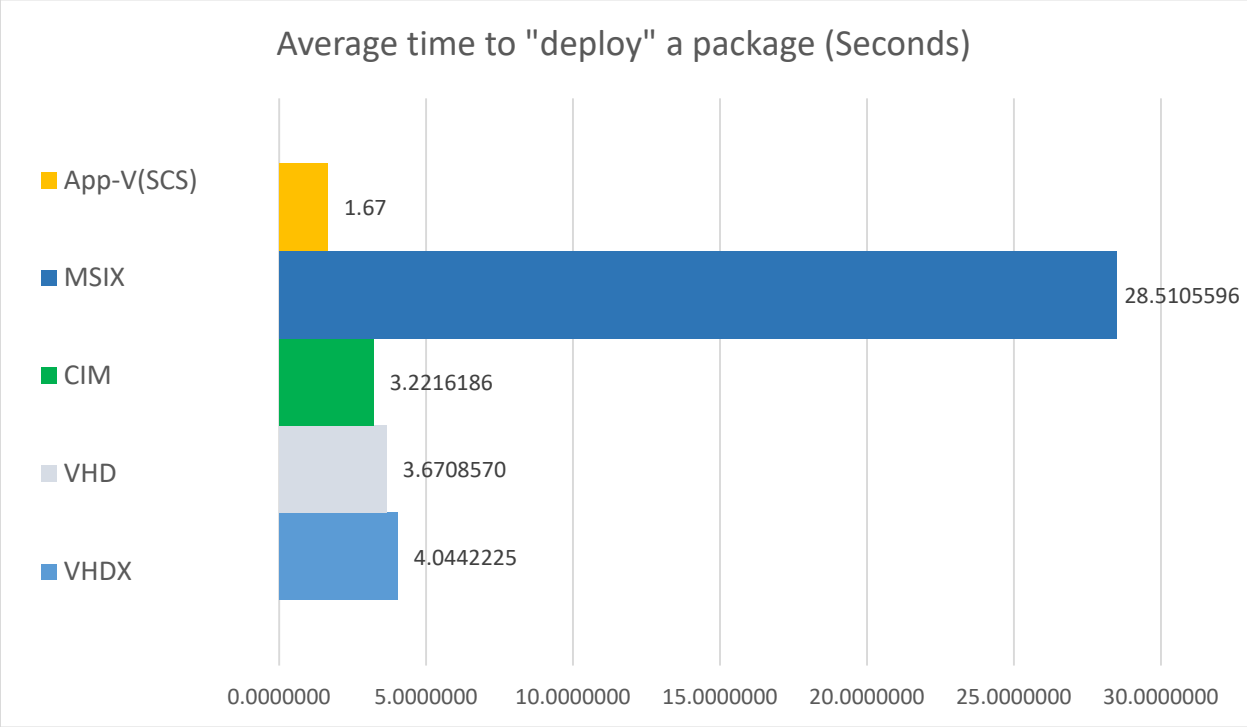
Format	Average Size (MB)
MSIX	306.5
VHD	1002
VHDX	1027.5
CIM	654.7

As these images are only mounted by, not copied to, the VMs the cost associated with this storage is minimalized, so this is more of a capacity issue.

Publishing Performance

When the user logs in, they may have to wait for their applications to complete a “publishing” operation before being available. As this occurs each time they log in, the publishing performance is of great importance. An average user probably has between 5 and 10 packages that must be published, and these occur serially.

We measured the publishing time for each of the different formats for a set of 90 packages, and the averages are shown in the chart that follows. We include here tests that also show the average time for these same packages using Microsoft App-V in Shared Content Store mode, and with straight-up MSIX installation for comparison.



We are not sure why App Attach takes longer than App-V, but possibly it is due to one or both of the following:

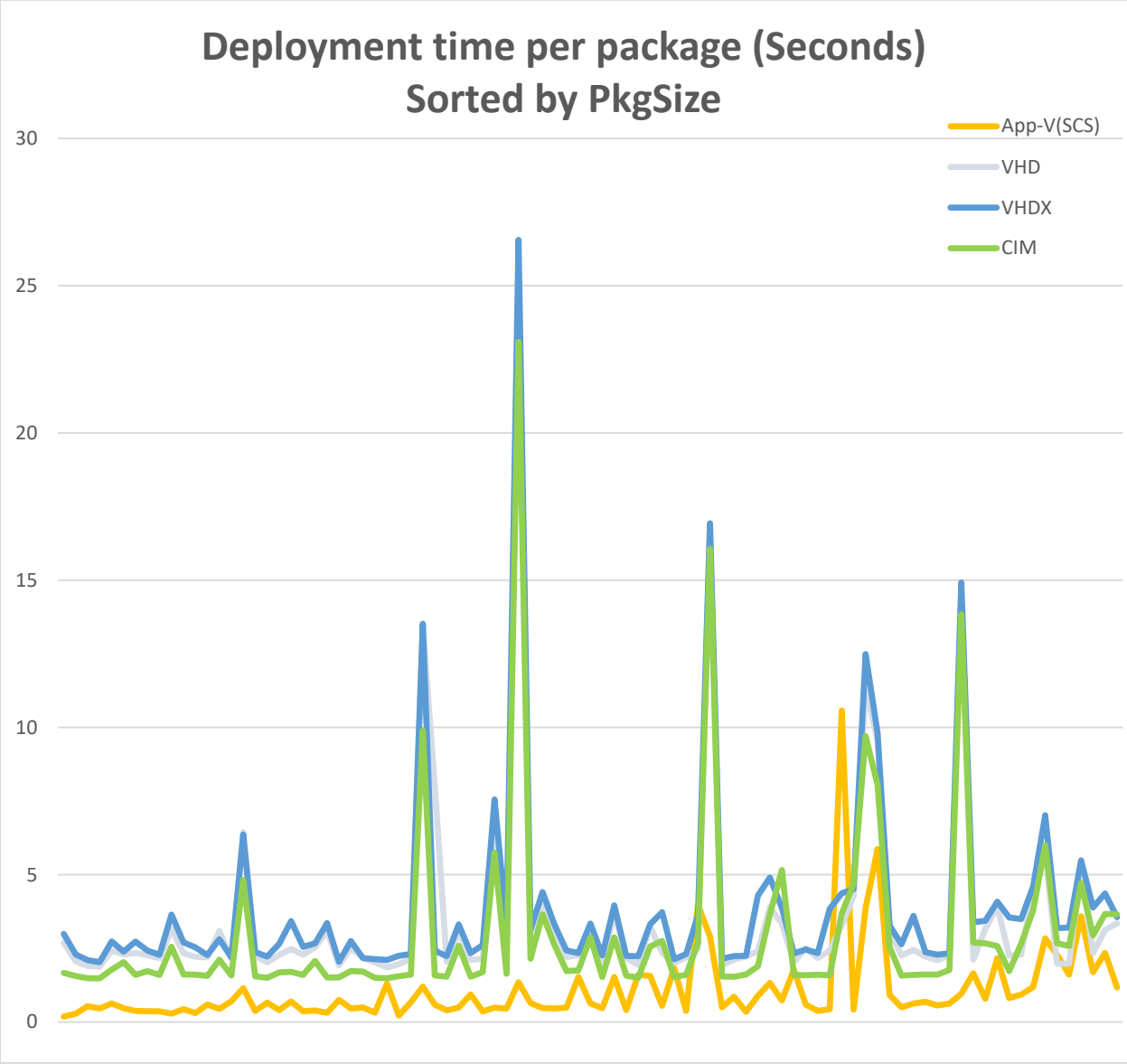
1. **File verification**. MSIX includes digital signature verification, and while verification of the entire package is lost in conversion to App Attach, some detail remains and we suspect that per-file hash verification might be performed during the staging operation.
2. **StateRepository**. Some storage operations during staging also require updating the internal StateRepository database on the VM. While some of this work is pushed to the background, it is single threaded and subsequent packages are affected by previously added packages. This can be seen in the GAP analysis of the logging.

Additional analysis of individual packages seems to indicate that while the length of time to publish a package has a casual relationship to the package size, other factors seem to have a larger effect:

- The presence of a windows service in the package.
- The number of files in the package

Although package dependencies (like Framework dependencies) are currently ignored and would have to be managed separately today, we expect that Microsoft may address this; if they do add this to the App Attach publishing this would also likely have a significant impact, even if the dependency is already present.

The following chart shows the publishing performance of the 90 packages on a per-package basis, sorted by the original package size (larger on the right).



Summary

App Attach is an effect means to deliver packages to non-persistent and semi-persistent operating systems. This avoids issues that tend to occur in larger organizations that must otherwise manage too many OS image formats with different combinations of applications and want the flexibility for quickly deploying up-to-date applications to users and avoid storage costs of individual customized images.

The application compatibility for App Attach is mostly the same as for that of MSIX in general, but there a small number of cases where App Attach may not work for a specific application package.