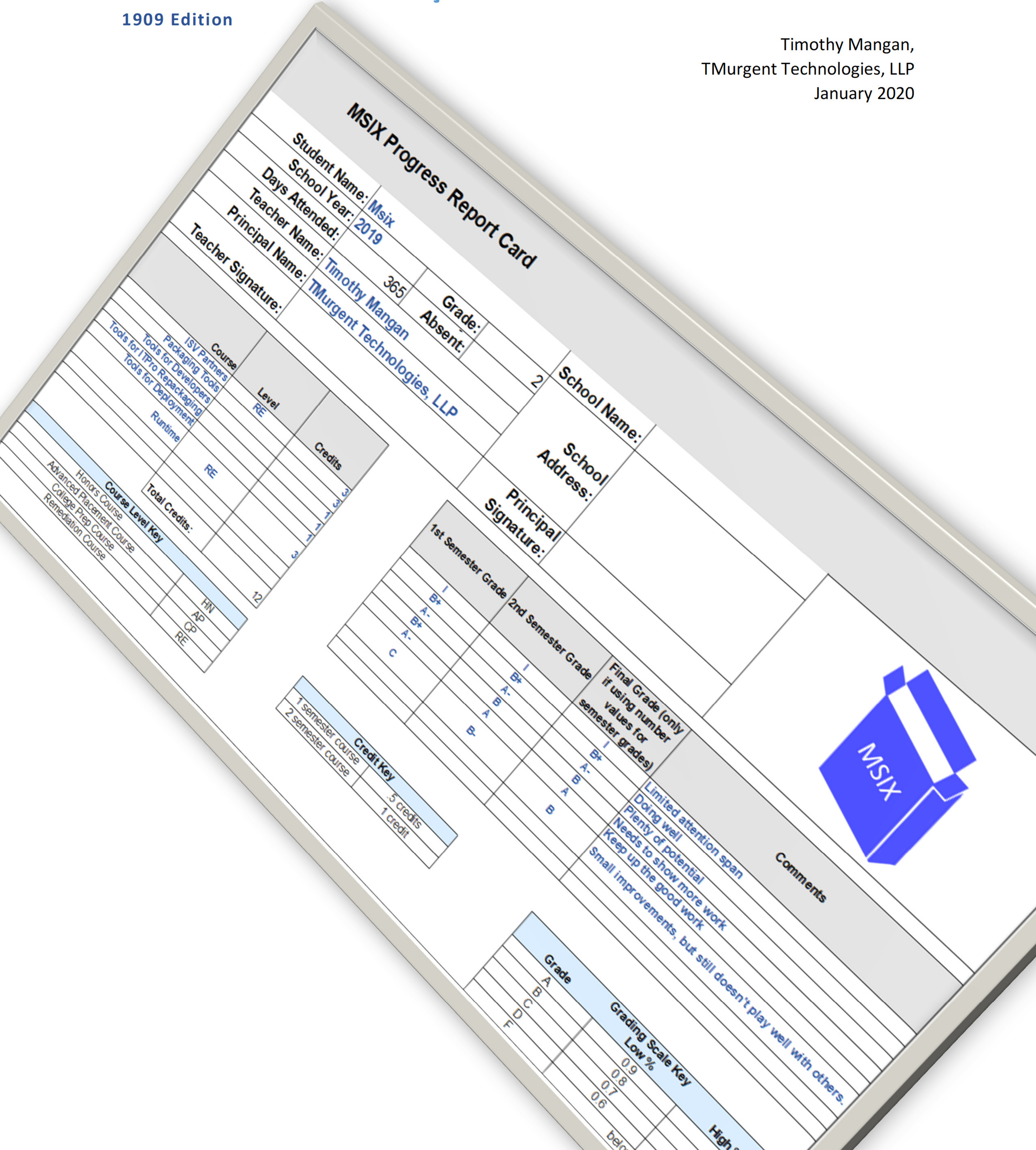
The MSIX Report Card

1909 Edition

Timothy Mangan,
TMurgent Technologies, LLP
January 2020



MSIX Progress Report Card

Student Name:	Msix		
School Year:	2019	Grade:	
Days Attended:	365	Absent:	2
Teacher Name:	Timothy Mangan	School Name:	
Principal Name:	TMurgent Technologies, LLP	School Address:	
Teacher Signature:		Principal Signature:	

Course	Level	Credits	1st Semester Grade	2nd Semester Grade	Final Grade (only if using number values for semester grades)	Comments
ISV Partners	RE	3	I	I	I	Limited attention span
Packaging Tools		3	B+	B+	B+	Doing well
Tools for Developers		1	A-	A-	A-	Plenty of potential
Tools for ITPro Repackaging		1	B+	B	B	Needs to show more work
Tools for Deployment		3	A-	A	A	Keep up the good work
Runtime	RE		C	B-	B	Small improvements, but still doesn't play well with others.
Total Credits:		12				

Course Level Key	
Honors Course	HN
Advanced Placement Course	AP
College Prep Course	CP
Remediation Course	RE

Credit Key	
1 semester course	.5 credits
2 semester course	1 credit

Grading Scale Key		
Grade	Low %	High
A	0.9	
B	0.8	
C	0.7	
D	0.6	
F	belo	

Introduction

Nearly two years ago Microsoft announced MSIX, a three headed effort that intends to be:

- A replacement for MSI based application delivery by software vendors.
- Tooling to help IT Professionals repackage existing software into MSIX. This involves both Microsoft developed tooling and tooling from third party vendors that are already in the repackaging business.
- And a runtime environment that embraces a modified form of the Microsoft Container used by Universal Windows Programs (UWP).

In the original announcements Microsoft indicated that the road to MSIX would be a journey and it would take several release cycles to complete all of the functionality needed. While some interpreted that to mean 6-month cycles I always assumed it meant 1-year cycles and openly suggested it might take more than that too. As 2019 ends and we start 2020, it is a great time to take another snapshot of how MSIX looks today. This *Report Card* is an update to our look one year ago (see http://m6conf.com/index.php/reportcard/46-report-card-2019)

We can summarize the changes as follows:

- Software vendors seem both interested but possibly wary, but, they generally have not made the leap yet. Microsoft's announcement on .Net Core UI3 has the potential to be the bridge they need; we will need to watch this space.
- Tooling for IT Pros is improving, but more is needed.
- With the Windows 10 1909 edition released last fall, Microsoft released the third version of the MSIX Runtime built directly into the OS. Complimenting this was backporting to versions of Windows 10 prior to the 1809 original release and MSIX Core, which could provide some functionality on even older OS versions like Windows 7.

For this year's *Report Card* I expand the depth of coverage over what was done a year ago.

- Significantly, we ran the first community surveys to gather opinions from folks in the field and those results will be reported in an attachment.
- The depth and width of testing for IT Pro repackaging has been expanded, with more applications, more in depth analysis, and more partner vendors involved.

This year the report is divided into four categories:

1. Support by Software Vendors to release in MSIX format
2. Support by Tooling Vendors
3. MSIX Runtime Support in the OS
4. Community Survey Results
5. Other options

My intent is to update this report card in January of the following years so that we can judge the progress. There are undoubtedly many other vendors that are active in this space and yet not included in this list due to my limited resources and/or unfamiliarity with their offerings. If you are one of these companies and are supporting MSIX, I apologize for the slight. Please contact me to ensure that I include you in the future.

And here is this year's report card…

1 Support by Software Vendor to release in MSIX format

I Microsoft's Independent Software Vendor (ISV) Partners, who build end-user applications for the Enterprise, earn another "I" (as in "Incomplete") for releasing software products in an MSIX format.

This is not a bad mark, but a reflection on where we are at this point. In last year's report, I said that I really did not expect to see many independent software vendors release in MSIX format until at least Windows 10 1903 was out, but more likely it would be after 1909, or possibly 2009, was released.

Vendors require a clear market, and clear set of tools that help them succeed, and a reason. They often only make big decisions on development plans only once a year. An established vendor wants to innovate and be leading edge but is not going to make the switch to something risky, at least not without a backup plan, and not unless they see value to them.

The market

The audience of customers able to accept MSIX apps is growing over time. Microsoft has taken many steps to improve this, including going back to make MSIX available on older Windows 10 versions (those still under mainstream support) and even the extraordinary release of what they now call "MSIX core" for other Operating Systems (more on that in a minute). And while this means that a vendor could theoretically release in MSIX form to pretty much everyone, there are practical limitations:

- Older OSs require a separate installation to support MSIX
- Older Oss don't have all of the latest capabilities of the MSIX runtime (and we aren't sure if they are more based on the 1809 runtime or something later). For example, Add-on packages either don't work or are impractical without the 1903 runtime changes. And Services and certain registry fixes are not due until 20H1 is out. Microsoft could be back-port these things, but we have not heard a commitment to do so.
- MSIX core doesn't have a runtime at all. It simply is a way to install the MSIX form in a way that acts like a MSI; the primary advantage is a single install format. If the vendor needed to make changes to run under MSIX, those changes might need to be removed to work in a MSIX Core environment. I'd rather just have the MSI in that case! Possibly there are other advantages for Microsoft and the developers with using MSIX Core on Ios/Android/Linux operating systems, but clearly these would not support the Win32/DotNet apps that enterprises are looking for.

The market for MSIX based apps continues to improve over time, but when is it enough for the vendors? Is this this year, is it the next?

The tools

The tools were largely in place last year, and they continue to expand and get better. I do not see this as a blocking factor for the software vendors.

The reason

Until recently, the MSIX story was more about benefits to others than to the software developers. Sure, it would allow them to bring their old code into the new format and customers would have better systems, but would the app run any better?

For some time an app had to choose between writing for UWP, using .Net Core, or for the traditional Win32 and .Net Framework routes. This decision affects both how the UI is written and the APIs available to them. And while UWP has some nice new UI features, the limited API made it hard to write enterprise scale apps. Microsoft is now addressing this and giving the software vendors a reason to want to move to MSIX. To overly simplify this, I will just call it ".Net Core UI 3" but basically it is a set of developer related tools that allow them to construct apps that contain a mixture of elements UI and API parts from both sides of the house. This is combined with the MSIX container to take advantage of it. This may become the reason for vendors to move, but only once they really understand what can and cannot be done with it. I suspect that like MSIX, Microsoft has more work to do in this area to convince developers that this is the way to go. But it feels like things are moving in the right direction.

The current state

It is quite difficult to know just how much MSIX is out there. You can tell if an app is in the UWP/Centennial/MSIX camp versus MSI/SetupExe camp easily enough, however, differentiating within that first camp requires digging into the internal manifest to find out. I have noticed that the MSIX Microsoft Packaging Tool is now MSIX (originally it was Centennial based), both Slack and Dropbox have MSIX apps, but not much else (outside of my own store apps).

Indeed, we can even question just what is an MSIX app once we start looking. For the purpose of this report, I am considering MSIX to be a package containing traditional Win32 or DotNet Framework components which requires declaring the restricted capability "RunFullTrust"in the package's AppXManifest file. There are about 100 UWP/MSIX apps that come delivered in the user's Windows 10 1909 experience out of the box, once logged in and the regional apps are applied. Most are UWP, 7 included the "RunFullTrust" capability[1], but there are some that use restricted capabilities only available because of MSIX, even if they do not include the "RunFullTrust" capability.

The likely venue to look for answers is the Microsoft Store apps, however it is impossible for anyone outside of Microsoft themselves to search the catalog in a meaningful way to determine which apps are UWP and which are MSIX. While I don't have any evidence to prove it, here is what I think: I don't think there is much MSIX released to the public at this point. I am aware that there are a few tools that folks involved with MSIX (such as myself) have released, and I suspect there are some MSIX games in the store as well.

And let's not forget about one of the biggest software vendors out there. What has Microsoft released? Outside of the Packaging tool and the 7 MSIX apps I identified as part of the OS, what else is there?. Certainly not the flagship Office, which would become a huge signal to the partner software vendors for the future of this technology. Will the next version of Office be MSIX based? We don't know. That is a

[1] Microsoft.Desktop.AppInstaller. Microsoft.MicrosoftOfficeHub, Microsoft.MixedReality.Portal, Microsoft.SkypeApp. Microsoft.Windows.Photos, Microsoft.XBoxGamingOverlay, Microsoft.YourPhone. If device is a Microsoft Surface, the Microsoft.SurfaceHub app is added.

huge undertaking! At least Microsoft already has experience in getting Office running under App-V, but running under MSIX brings in lots of additional challenges. Perhaps they'll have to settle on parts of Office in MSIX, and part not. If we assume that the next Office release will be a 2021 version, then we should start to see signs on their approach later this year as preview become available.

Another place to look is activity in forums where developers ask questions and I see little sign activity there outside of those just starting to kick the tires.

I assume that Microsoft is working closely with a few of the top tier developer partners, and those activities will help shape the development of guidance to others, but the publicly available guidance that I have seen so far have not shown a lot of depth.

While I did conduct a survey for developers to try to get their opinions, it is not clear whether it simply failed to get to the target market of those devs involved or whether there just aren't too many out there yet.

While the grade for software developers remains at Incomplete this year, it is a more nervous Incomplete than a year ago. My hope is that we see a few big successful releases this year, leading to a wider adoption in 2021. My fear is that next year I still won't know. It's all about the apps; this Report Card starts with the software vendors first because of that. If MSIX is going to succeed with MSIX, vendors have to release in MSIX format.

2 Support by Tooling Vendors

B+ Tooling Vendors stayed steady at "B+" this year. The vendors, especially the most popular vendors, have been very active in the MSIX community.

This category is further broken into three sub-categories, and I will grade each of these independently as well. The category includes:

- Tooling for developers
- Tooling for IT Pros in repackaging
- Tooling for distribution.

Many of the players are involved in multiple of these subcategories, and Microsoft themselves are also a first party vendor in this space as well.

But as it is an evolving technology, without a well-defined public roadmap, it is difficult for third parties to judge when and how much effort to put into MSIX. There is a very real possibility that work done to have product today will have to be reworked or discarded as Microsoft makes changes to the runtime in the future.

2.1 Tooling Vendor Support for Developers

A- Tooling Vendors supporting Developers earned an "A-". Many of the vendors have products available and you can build and deliver applications using these, as long as the applications do not run afoul of the MSIX Runtime restrictions. Last year vendors earned an "A-", and I suggested that these vendors will need to up their games to earn such a high mark. While I have not seen significant improvements, there is just enough to maintain this mark.

Most all of the traditional installer tooling vendors support building MSIX packages. Several new companies have also sprouted up in the last couple of years as well. These challengers are offering generally simpler products at a lower price, but completion in any form tends to make everyone work just a little harder.

In addition to that tooling, there is the open source MSIX Packaging SDK on Github. While I haven't noticed anyone turn that into a prebuilt functional equivalent to free open source install builders (like the popular free NullSoft installer for traditional setup packages), there probably is someone working on it somewhere. Supporting the open source community with pre-built tooling is needed.

Going forward, all vendors will need to consider how to help developers create applications using a mixture of traditional and modern components. It isn't necessarily enough to say you can have both in the package, the tooling must assist in the process.

2.2 Tooling Vendor Support for Repackaging

B- Software Vendors supporting IT Pro Repackaging of applications into MSIX earned a "B-" this year. The initial *Report Card* talked about the great initial efforts in this space, well ahead of what was happening in the developer space. Progress has been made this year in tooling, but we still need to wait for improvements to the MSIX Runtime, the AppXManifest schemas, and Package Support Framework.

Microsoft delivered multiple releases of the Microsoft MSIX Packaging Tool (MMPT), reacting to feedback from the early version we were testing a year ago. It is a bit more mature and easier to use than a year ago. Significantly, my testing indicates that issues with creating some form of package have largely been eliminated, but there are still many API features used by traditional apps that should be able to work under the MSIX runtime that it does not support.

Meanwhile, the Package Support Framework (PSF) has become a reality. This is an open source project hosted on GitHub that includes Detours (some of the technology behind App-V and other vendor products) for Windows API interceptions. IT Pros don't want to build their own C++ code to fix up their repackaged apps, so having pre-built versions is important. One form of this is PsfTooling, a free app in the Microsoft Store that can be used while in any installation monitoring tool (like the MMPT) to inject and configure the components. Other packaging vendors also include the PSF in their own products. The issues with external submissions into the PSF source that were experienced last year have improved and the supported functionality has expanded because of this. Considerable improvements have been made to the PSF that overcome certain limitations of the MSIX Runtime, and a more flexible approach to how redirection occurs was added to support roaming user scenarios. My testing reflects an increase in the percentage of applications that are compatible for repackaging into MSIX as an outcome of these changes.

Third Party vendors that offer repackaging tools continue to work on their products. Flexera, which a year ago had not yet released an Admin Studio with MSIX output capabilities now has. While these vendors generally hopped on the MSIX bandwagon early, my sense is that they have experienced less than expected customer demand, and until MSIX matures they will likely place more emphasis on other products/features until then. The testing performed as part of the *Report Card* for this year shows that these tools have work to do just to catch up with Microsoft. These tools should produce better results than the incumbent, or at least equivalent results with a superior experience and/or integration with other packaging related tasks.

2.3 Tooling for Distribution

A Software Vendors with support for MSIX Distribution at a solid "A". I'd give it an A+, but until we have production ready packages to distribute, there isn't even a point in testing what we have gotten.

Microsoft themselves have support for MSIX in the Microsoft Store (the "Consumer Store), the Microsoft Store for Business, Intune, and SCCM. Additionally, we can use the same PowerShell commands used to install and uninstall AppX (UWP based) and Windows Bridges (Centennial based) programs.

This pretty much means any vendor using the AppX PowerShell commands can probably handle MSIX without any changes, with the possible exception of understanding the new file extension.

A long awaited entry into this space is also from Microsoft; *MSIX App Attach* was released at the end of the year as a way to support MSIX in a non-persistent Windows Virtual Desktop environment. Built by the folks they acquired from FsLogix, it offers a fast installment of MSIX apps for scenarios where the user logs into a “fresh” copy of Windows each day and dynamically delivers assigned apps on the fly. This type of support is critical if MSIX is to replace App-V in the non-persistent and semi-persistent environments. While the demos look good, the release came so late so as to be untested in this *Report Card.* One concern I do have is lack of full feature support for MSIX when using MSIX AppAttach to deploy. For example, we know that MSIX is expected to start supporting packages with Services in the first half of 2020, but indications are that this is architecturally incompatible with how MSIX AppAttach works. We know from our work in App-V that this will be a blocking factor for the enterprise if not addressed down the road.

3 MSIX Runtime Support

C The MSIX Runtime moved up to a “C” grade this year, up from a “C+” last year. Ultimately what is important is what can I deploy into production now, so progress has been made but we still have a long way to go in App Compat.

The significant improvements to the MSIX runtime that I noted this year include:

- *Support for Modification Packages (Add-Ons).* The 1903 OS release included fixes to the VFS layering that now make it possible to use Add-On packages that replace (update) files in the main package. This would allow, for example, a vendor to release their product with a default configuration that can be changed by Enterprise IT without having to repackage the main package at all. Unfortunately, this support has not been back-ported (to my knowledge), and only works if the files in question are packaged using VFS pathing.
- *FTA, Shell Extensions, and Protocol Handlers* The 1903 OS also included improved support for packages using these types of integrations. There are forms of each of these that still do not work, and Microsoft is expected to expand this support over time.
- *File Access Changes.* I don’t know when these changes occurred, but when testing on OS 1909 with December 2019 feature and security updates for the OS, I detected a change in file access that allows some apps to now work without the use of the PSF when previously the addition of the PSF FileRedirectionFixup was necessary. There appears to be no release note information on a change, but if an app uses a particular method to open a non-VFS file for write access it will be automatically copied into the user profile and writes will be performed there (previously the app would receive an access denial). This is only one of many forms of file access that we’d like to see addressed in the Runtime; it will be interesting to learn if this is a trend we should expect or an exception to the previous explanations that we would need to use the PSF to sort out these compatibility issues.

The list of things not yet supported remains large. We would have liked to see more done in the 1909 OS runtime. Indeed, I expected that Services would have made it into 1909, but it looks like that will be in the 2020H1 version of the OS.

Enterprises depend on a rich variety of Windows APIs that provide integrations with the OS and other applications, enabling not only application features, but also the enterprise to build a user-centric workflow that involves multiple applications to complete the task at hand. While the current MSIX Runtime container is more capable than the UWP or Centennial containers, it is still a container that isolates the application from other applications and the OS itself. Uncertainty if MSIX will become capable of running all the apps that Enterprises count on still exists.

Microsoft spent a decade solving issues with isolation via App-V, but the fresh approach of MSIX seems to require re-inventing the wheel for lessons learned in the past. Microsoft is clearly working on filling these gaps, but information about this is limited. A roadmap detailing what will and will not be addressed and when would be very helpful to all involved. In today’s era of shareholder lawsuits companies don’t talk about future plans, but disclosures of intent can be stated without making commitments can be achieved.

4 Repackaging Testing

At the end of the day, we need to be able to deploy the apps. Without a log of vendor supplied apps available, we can work the path of repackaging existing applications into MSIX and testing those. In this section, I'll document the result of the testing. The experience with this testing was crucial to, and had significant impact up, the grading earlier in this report.

This year, as part of the *Report Card*, I expanded the set of applications to be tested in a repackaging scenario to 60 . The majority of these applications were applications that we commonly see Enterprises distributing to employee workstations today. A few "special purpose" applications that I commonly use to demonstrate application integration capabilities were also included to ensure that we are covering the needs of most apps. Not included in the application mix were applications that are known to not work under any virtualization or container, such as App-V and MSIX – for example those with device drivers or plug-ins for Internet Explorer. Also excluded were problematic "heavy" applications like AutoCAD and ArcGIS that are always difficult to deploy.

These applications were repackaged and tested six times:

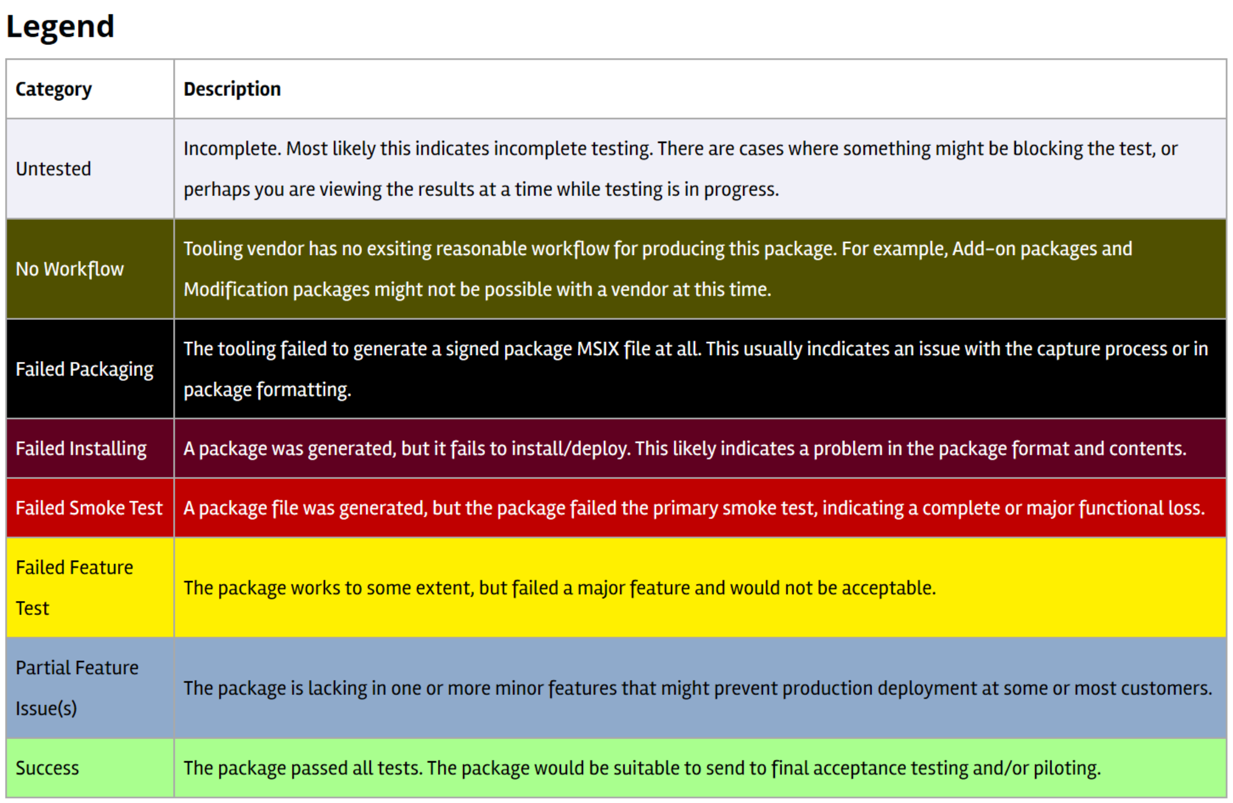
- Using Microsoft App-V
- Using the Microsoft MSIX Packaging Tool (2019.1018 release) without shimming
- Using the Microsoft MSIX Packaging Tool (2019.1018 release) with PsfTooling 3.2.0
- Using Advanced Installer Architect (16.7) with their version of the PSF
- Using Flexera Admin Studio (2019R2) with their own launcher.

For each application, I categorized the results of testing into one of five buckets:

- **No Packaging Workflow.** The tooling does not yet support a way to package the application. Right now, Add-on packages are a problem for some of these tools.
- **Does Not Package.** Using all the tricks that I am aware of, I could not get the app to produce a MSIX package file. This category includes situation where an MSIX file would get created but it could not be signed by signtool.
- **Failed Installation.** A package was created and signed, but AppInstaller refuses to install it.
- **Failed Smoke Test.** A smoke test is nothing more than installing the package and trying to see if it installs and the primary application shortcut can be launched. For apps that are simple, the primary use of the app might also be tested. Passing the smoke test does not mean that the application is production ready, only that it is good enough for full acceptance testing.
- **Failed Feature Test.** The acceptance test showed a failure that would clearly prevent an enterprise from putting this package into production.
- **Partial Feature.** The acceptance test showed that the most important features of the application work, but that some features were not available. A subjective decision was made that the lack of the feature(s) would keep most enterprises from releasing this package into production, but that some might. In addition to features not, and app can fall into this bucket due to the inability to disable updaters or application licensing challenges.
- **Full Feature.** While not every feature was necessarily tested, the Acceptance test indicated that it is highly likely the package could be put into production.

While every attempt is made to be objective in testing, the interpretation of test outcomes nevertheless is subjective. Someone else testing the same packages might categorize the results into different buckets than me (especially between the last two buckets). But we have to start somewhere!

In the charts that follow, color coding is used to signify the categorization of the testing result. The following table should be used to interpret those colors:



Legend

Category	Description
Untested	Incomplete. Most likely this indicates incomplete testing. There are cases where something might be blocking the test, or perhaps you are viewing the results at a time while testing is in progress.
No Workflow	Tooling vendor has no exsiting reasonable workflow for producing this package. For example, Add-on packages and Modification packages might not be possible with a vendor at this time.
Failed Packaging	The tooling failed to generate a signed package MSIX file at all. This usually incdicates an issue with the capture process or in package formatting.
Failed Installing	A package was generated, but it fails to install/deploy. This likely indicates a problem in the package format and contents.
Failed Smoke Test	A package file was generated, but the package failed the primary smoke test, indicating a complete or major functional loss.
Failed Feature Test	The package works to some extent, but failed a major feature and would not be acceptable.
Partial Feature Issue(s)	The package is lacking in one or more minor features that might prevent production deployment at some or most customers.
Success	The package passed all tests. The package would be suitable to send to final acceptance testing and/or piloting.

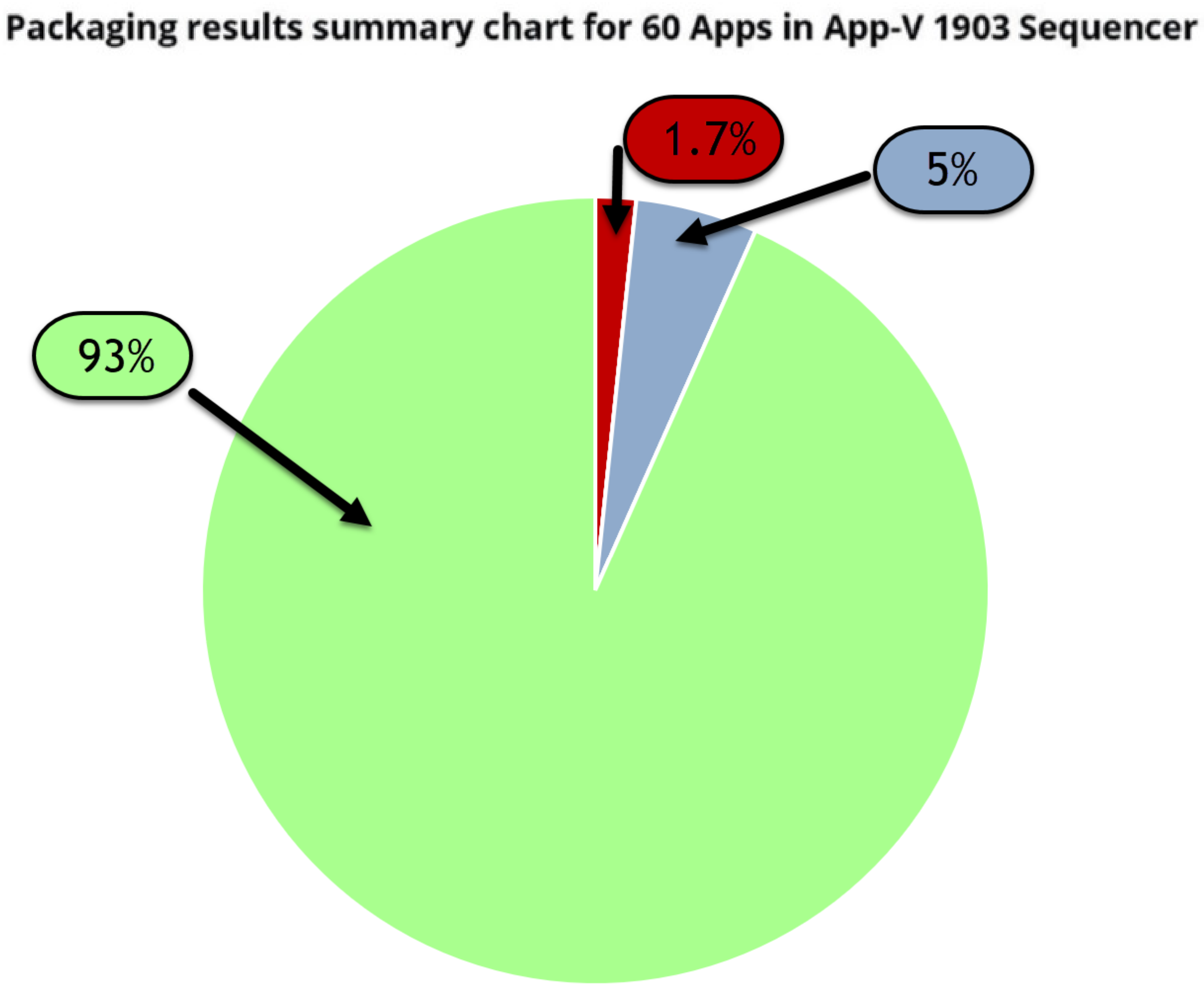
Finally, it is worth noting that the testing performed was using recapturing techniques. There may be ways to build a MSIX package using traditional install builder tools from these same vendors that have been extended to support MSIX, however that was outside the scope of this testing.

4.1 Comparative results using Microsoft App-V

Many of the applications in the list are older applications that are challenging to deploy in today's environments. Indeed, although I did not categorize the list using Native installation techniques, the results would be far less than for repackaging in App-V. This is after all why we have App-V!

The 60 packages were packaged on Windows 10 1903 and tested on the same OS. Due to a long-standing bug on Short-Names in the Sequencer that appeared in the 1809 release and has not been addressed, many customers are continuing to use the 1803 Sequencer. I used a mixture of both of the common approaches being used by App-V customers today:

- The ADK for 1803 Sequencer
- The ADK for 1903 Sequencer (a 1909 ADK is not forthcoming) supplemented by my TMEdit program to post-process the package to solve the Short-Name issue.

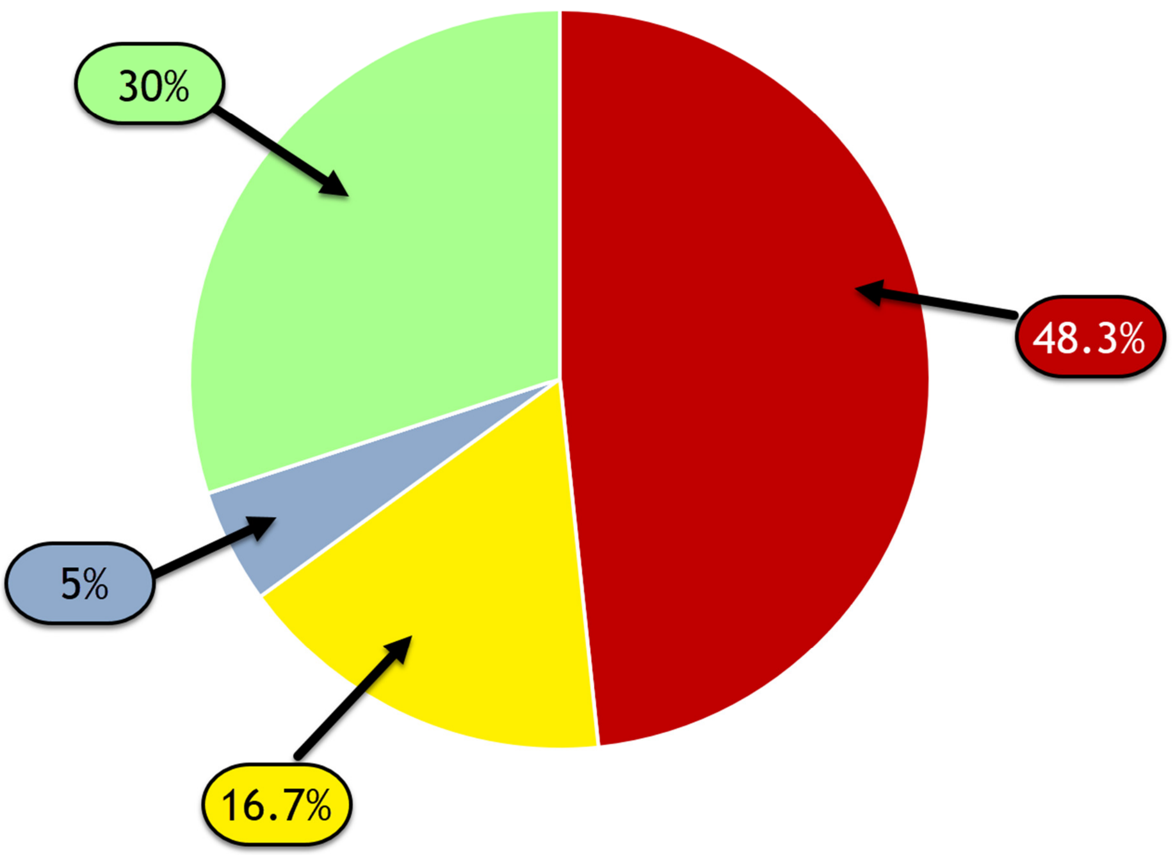


Not surprisingly, these results were very good using seasoned tooling and collective wisdom on techniques to package with App-V. If anything, I was harsh in the objective judgment on those 4 apps.

4.2 Results from Packaging with Microsoft MSIX Packaging Tool

The same 60 applications were packaging using the 2019.1018 release of the MMPT.

Packaging results summary chart for 60 Apps in MMPT 2019.1018



This is a significant improvement of the tests from last year. Last year we were testing against the initial public release of the Packaging tool, and had many challenges just in getting packages created.

- This year we suffered no failures in creating the package, an improvement that is strictly affected by improvements to the tool itself.
- The increase in Acceptable packages from 22% to 30% this year is due to a combination of better recapturing on the part of the MMPT and improvements in the MSIX Runtime.

These improvements are encouraging, but not likely to convince an enterprise that MSIX is ready yet for their apps, at least not without additional help.

The MMPT includes a remote option now, however it assumes that you use an external tool to control the state of the capture virtual machine, so I did not use it. It seems appropriate only for a packaging house that builds their own tools.

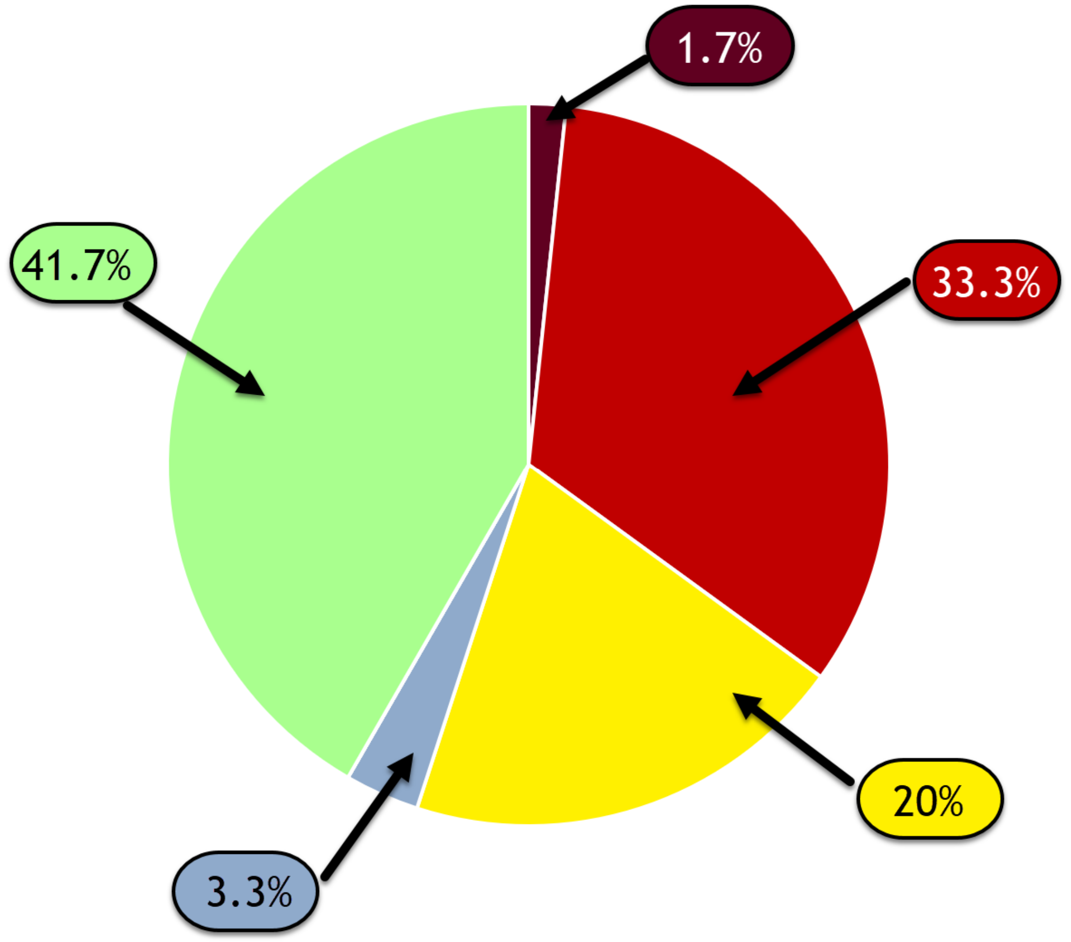
4.3 Results from Packaging with Microsoft MSIX Packaging Tool with PSF

For these tests, the MMPT version 2019.1018 was also used, but the packaging process was enhanced by using PsfTooling 3.2.0 to inject and configure the Package Support Framework (PSF) into the package. This version of the PSF contains a build of the latest PSF source code from GitHub available in December of 2019. The same set of 60 Apps were used, but not all were repackaged:

- 28 Packages contained issues that we known to be addressed by the PSF.
- 32 Packages either had no issue requiring the PSF or the PSF had no available fixes.

Of course, just because the PSF has a fix for an identified issue doesn't ensure an acceptable package; many packages contain multiple issues and the PSF might only fix a subset of them.

Packaging results summary chart for 60 Apps in MMPT with PsfTooling 3.2



Last year I excluded the PSF from *Report Card* testing with the MMPT because while *it existed, it was* only available in source form and IT Pros were not going to build their own tool.

The primary impact in results comes from application of the FileRedirectionFixup. Using this fixup solves issues with applications such as:

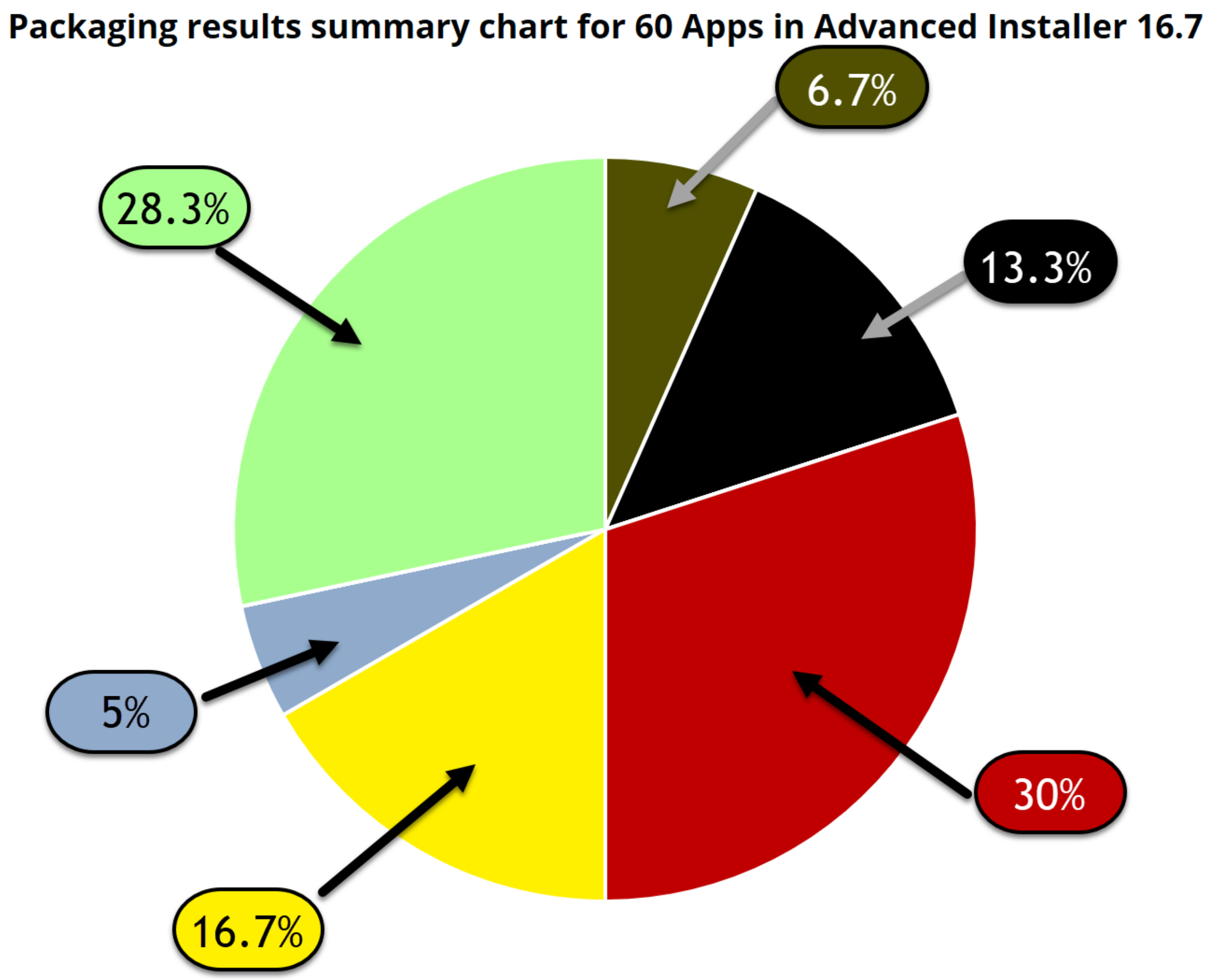
- Not being able to see configuration and plug-in binaries placed in the VFS equivalent of the user AppData/Local and AppData/Roaming folders.
- Not being able to write to configuration files anywhere in the package.

Additionally, the new DynamicLibrary fixup that solves issues with applications find it's dlls in the package due to a lack of support for Path variable modification or support for AppPath registration.

Not affecting the measured results, but equally important were changes to allow configuration of the redirection destination when using the PSF. This change enables the acceptable apps to be deployed to less static scenarios, such as non-persistent VDI, semi-persistent environments like RDS, Citrix, and VMWare Horizon, scenarios involving additional products like UEV and Ivanti to handle user data, and desktop replacement strategies that depend upon all use data being redirected to an external home drive.

4.4 Results from Packaging with Advanced Installer (16.7)

The same 60 apps in the prior test are used in these results with a pre-release version of Advanced Installer 16.7 from Caphyon. The pre-release was used because version 16.7 was released prior to the publishing of this report.



As was the case last year, Caphyon still has a few issues with the capture for certain packages. I share details of the issues found with the vendors as part of the testing process, and I expect that many will be addressed shortly.

When I shared these results, the folks at Caphyon showed me that the VFS overlay modification packages are possible by avoiding the *Advanced Repackager* tool and instead creating the package directly in *Advanced Installer*. This install builder technique works only as long as there is no installer to run, in other words as long as you are just copying files. Using these techniques instead should make packages that landed in the "No Workflow" result category possible, and might avoid issues that caused apps to fall into the "Failed packaging" category.

Caphyon, like Flexera and several other of the third party tooling vendors, use a recapture technique that enables the capture to be used in a variety of formats, including MSI, App-V, and MSIX repackaging. The capture device can be a remote VM and they manage the entire process. The shared recapturing tooling is a blessing, as the products are more robust and may already be familiar to the packager, but can also be confusing to new users due to interface options not being designed for the one thing that you are trying to do.

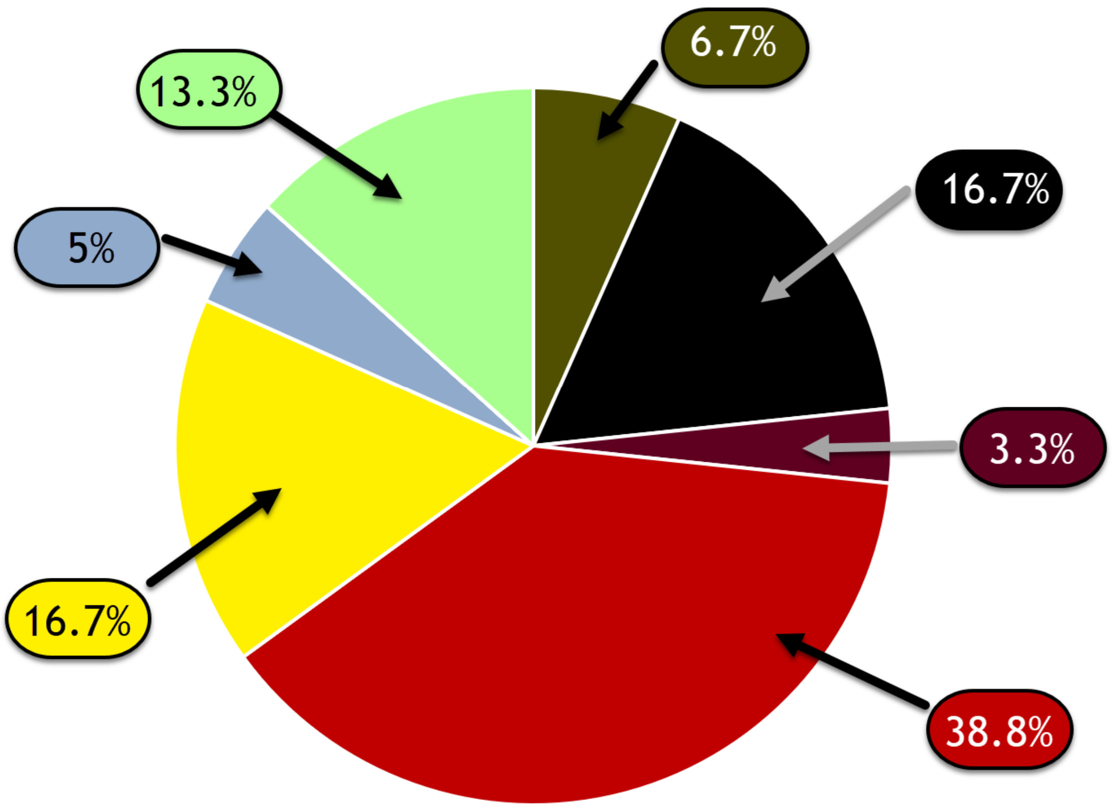
Caphyon includes their own private fork of the PSF source code. This helps them in some cases, as they have created fixes not shared with the original source but hurts them in some of these tests as had not yet incorporated the latest changes to the PSF prior to their product build. Their user interface for injecting and configuring the PSF is less intimidating than that of PsfTooling, but at the loss of a little bit of configuration flexibility. The loss of flexibility did not affect any of the testing outcomes here, except for the packages marked “No Workflow”, however more manual work was required to get the application icons in place.

Currently, Caphyon does not have a workflow through the *Advanced Repackager* tool that would enabled me to create the equivalent of Modification Packages in the MMPT for use with VFS pathing. This has been a catch-22. You need to use VFS pathing to do modification packages that replace/update files in the main package, but the PSF was designed for packages at the root of the package (equivalent to PVAD style packaging in App-V). As this was only added to the PSF in December, Caphyon will likely enable these workflows in the future. This would allow for independent packaging of plus-ins and potentially independent packaging of application configuration files (the latter scenario is not yet included in the tested application mix).

4.5 Admin Studio

Admin Studio 2019R2 was also tested against the same 60 applications in a recapturing test.

Packaging results summary chart for 60 Apps in Admin Studio 2019R2



I couldn't test Admin Studio last year, as Flexera's support for MSIX was for developers and analysis of MSI packages for MSIX compatibility only. So this marks their debut in the testing. I did perform some preliminary testing on their first version supporting MSIX repackaging, and the results using R2 are remarkably better.

Admin Studio has a fully managed remote capture capability which is also shared with repackaging for MSI, App-V, and MSIX.

While the InstallShield side of the house (used by developers to build packages from source) does have PSF capabilities, this appears to not have been added to Admin Studio for recapturing yet. If you package with an evaluation version it injects their own launcher program so you can see how the process might work when they do. I did try to use PsfTooling inside their recapture, but unfortunately they create the AppXManifest file differently than the MMPT so this will not work today without extensive manual labor. Basically, with Admin Studio today I was unable to fix any of the problems that I'd normally fix using the PSF so a comparison to the MMPT without fixups might be more appropriate.

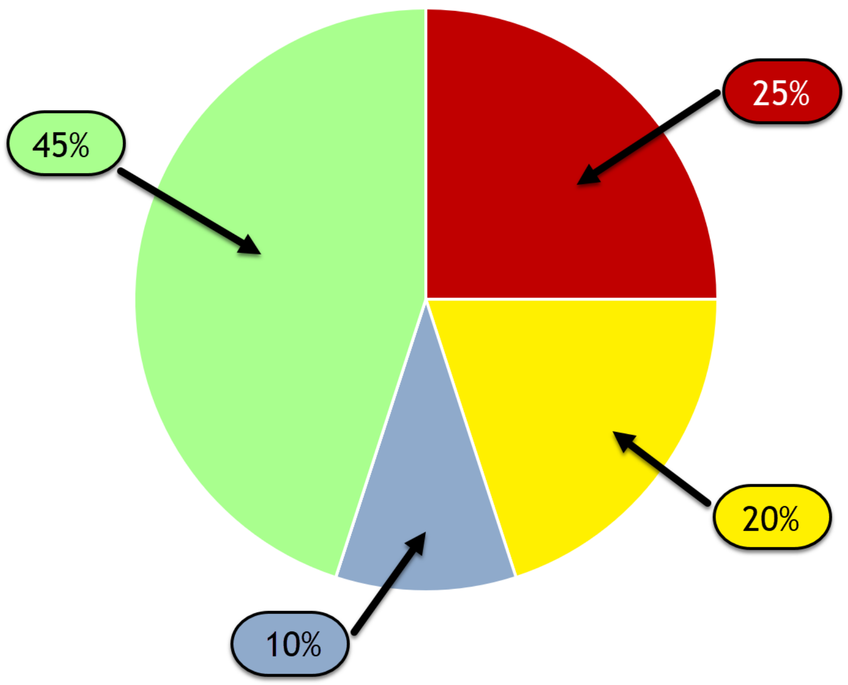
Like Caphyon, there are still some capturing issues and a lack of workflow for those plug-ins in Admin Studio. It likely is also possible that by using InstallShield for situations where an installer doesn't need to be run, or is an MSI, a workable solution is possible, but this style of package creation was not included in the testing as IT Pros do not generally use InstallShield.

Admin Studio also offers the ability to work with an imported MSI based installer without using a recapture phase but, as with other vendors, that was not attempted in this testing. This might allow one to avoid some of the recapture issues that caused failed packaging, failed installing, and failed smoke tests and lead to improved results.

4.6 Best Case Scenario for MSIX Packaging

The following chart represents the “best case scenario” for handling the applications; either any of the five recapturing scenarios to product MSIX packages (i.e. excluding the App-V scenario) reported previously in this paper. This chart has the “Best Result” for each of the 60 applications tested this year.

Packaging results summary chart for 60 Apps using best results (skipping first scenario result)



This is a big improvement over last year. But yes, we still have a long way to go. An enterprise isn’t going to package using three different tools, and they need a higher acceptable app-compat.

A chunk of this year’s improvement came from addressing single issues that affected many applications. Going forward each fix added, whether it be to the Runtime, the PSF, or the Tools, will have less of an impact on results.

However, on the flip side we have already seen pre-release software (not included in this testing) for the MMPT and the 2020H1 OS that address significant issues associated with Services and the Registry. The latter is a huge relief to be because it is high on my list of things I thought I might have to try to add to the PSF. That Microsoft fixes it in the OS Runtime is a huge relief as that is where it belongs.

What’s missing in the MSIX 1909 Runtime

My short list of support missing in 1909 includes the following two lists. The first list is potentially fixable by existing or potentially developed shims, the second I believe can only be addressed in the MSIX Runtime. There are more items that should be added to these lists, but these are the what I believe to be the key ones.

Currently fixable via Packaging and Shimming:

- Ability for the package to contain user modifiable configuration files.
- Application Settings and Data in the package AppData Local and Roaming folders not seen by the application without shimming.
- Shortcuts that have arguments.
- Shortcuts needing working directory.
- Shortcuts to files
- Path variable and AppPaths for dll search not supported.

Known to be expected in 20H1:

- Support for Services.
- Support for apps modifying registry items defined in the package.

Should require a new MSIX Runtime, and often MSIX format changes:

- Fonts in the package are not supported.
- Environment Variables are not supported.
- Support for plug-ins for natively installed apps.
- All Shell Extensions (some forms are supported today
- WMI Providers
- ETW Provider Formatters
- Windows Scheduled Tasks and Run keys
- Software Client System (default browser or mail client support, for example)
- Exposing COM objects
- Spoofing for Named Kernel Objects (oddly included in App-V and Appx but not MSIX)
- Layer hiding (Override Local in App-V)
- Deletion Markers for file and registry systems (used for things like multiple Java runtimes)

There are more items that should be added to these lists, but these are the what I believe to be the key ones. Microsoft has committed to addressing some of these (no hard dates) and has publicly acknowledged looking into others. The remaining set they have not publicly commented on (to my knowledge), but at least they haven’t said “no” to.

5 Community Surveys

This year I created the first two Community Surveys to to explore the public opinions about how they view MSIX. The intent of these surveys was both to provide the community a chance to see what their peers think, and to provide Microsoft an additional avenue of feedback.

Two surveys were conducted. The first was intended for people at companies that produce software for internal or external use. The second was for people at companies that acquire and distribute applications internally. The full details on the surveys are available as a separate download from the https://www.m6conf.com website.

The survey includes questions regarding participant background, experiences, opinions, and plans regarding MSIX.

About the Author

Tim Mangan is an independent consultant and the owner of TMurgent Technologies, LLP. Recognized as an industry leader by Microsoft as an MVP for more than a dozen years, and by Citrix for a half dozen as a CTP Fellow, Tim is generally known as "the Godfather of App-V", having led the effort to build the original version of App-V at Softricity.

TMurgent provides consulting and training around application and desktop deployments. Our "Packaging for App-V and MSIX" training classes are sought after by IT Professional Desktop Engineers around the world.

TMurgent Technologies, LLP is an independent company engaged in the packaging space. TMurgent primarily provides training to IT Professionals that are involved in Desktop Engineering and Application Packaging, but we also provide some free and licensed software products in this space.

As an independent contractor, we may relationships with several of the vendors, some of which provide support. Despite this, we believe that the information in this report does provide a fair and independent view that represents the state of the industry at this time.

This report contains information gained from personal experiences and may not represent the best that can be said about the vendors and products mentioned. Omissions and mistakes are my own, but they are "honest" mistakes and not intended to malign.

Vendors have not been given an opportunity to review or correct potential factual errors in this report. We hope to be able to do so with any future reports that we do.