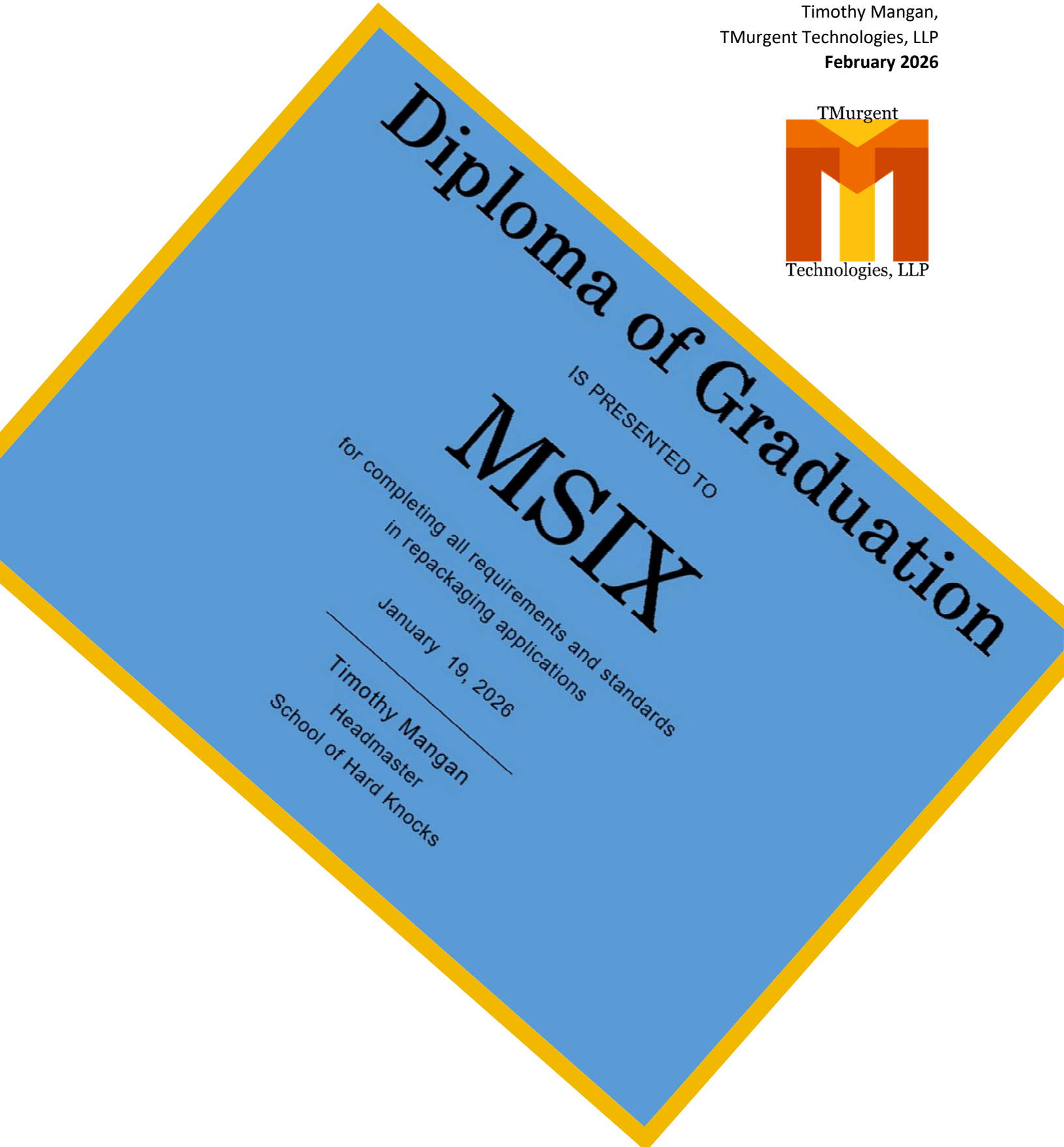
The (final?) MSIX Report Card

2026 Edition

Timothy Mangan,
TMurgent Technologies, LLP
February 2026

TMurgent Technologies, LLP



Introduction

This is the 8th, and possibly final, Report Card on MSIX. An annual report that I have produced on the state of MSIX. As indicated in the report, MSIX has advanced to the point as to no longer need the detailed kind of testing on app compat.

When Microsoft announced MSIX many years back, they described a four headed effort that intends to be:

- A replacement for MSI based application delivery by software vendors.
- A new SDK for developers to use. The Windows App SDK (formerly known as Project Reunion) helps the developer to write natural code that works inside the MSIX container, or now also possibly outside of the container as desired.
- 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 protects applications and the operating system by using an updated container similar to that used previously for UWP programs.

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 the functionality needed. For some years customers have become outspoken about the issues they encounter and question just how committed Microsoft is to this path.

While the journey has been long, as you will see in this report, MSIX is ready.

This 2026 version of *The Report Card* is an update to our prior views; to review previous versions see:

- http://m6conf.com/index.php/reportcard/46-report-card-2019
- http://m6conf.com/index.php/reportcard/46-report-card-2020
- http://m6conf.com/index.php/reportcard/46-report-card-2021
- http://m6conf.com/index.php/reportcard/46-report-card-2022
- http://m6conf.com/index.php/reportcard/46-report-card-2023
- http://m6conf.com/index.php/reportcard/46-report-card-2024 and
- http://m6conf.com/index.php/reportcard/46-report-card-2025

I am uncertain if this will be the last year for The Report Card. My intent was to provide information on MSIX, but in particular as to the level of application compatibility that can be achieved. As that level is now at least as good as other packaging options available to organizations, this might be the last report.

Going forward the focus will be turning. Efforts to do better on App Compat will go on, but now the emphasis will be on other blockers for the enterprise in using this technology.

And here is this year's report card…

Executive Summary

We can summarize the changes this year as follows:

- Migrations to Windows 11 are now complete for almost all organizations. Windows 11, and Server 2025 serve as the baseline that supports so many of the Extensions that Microsoft has added to MSIX. While Server 2022 to 2025 Migrations are still going on, we now have a good base of features to work with in operating systems in production.
- Research into some of the under-documented portions of MSIX this year has opened up new possibilities that we did not have a year ago. This includes:
    - Additional controls over File System Virtualization. See Flexible Virtualization (FileSystemWriteVirtualization).
    - Additional controls over Registry Virtualization. See Flexible Virtualization (RegistryWriteVirtualization).
    - Windows Firewall Rules. See Windows Firewall Rules and MSIX Packages.
- The release of *TMEditX Tools* as a free community toolset to help with the testing of MSIX packages. These tools simplify the process of unit testing and debugging packages prior to UAT. Other "App Lifecycle" vendors are expanding their capabilities, most integrating TMEditX into their workflows.
- Added additional application compatibility for repackaging apps using an enhanced *TMEditX Editor*.
- An uptick in the number of 3rd-party vendors that have releases already in MSIX, and continued expansion by Microsoft itself.
- The arrival of the "hybrid installer".
- A surge in organizations adopting MSIX, in particular due to the end-of-life of the App-V Server. Many have turned to AppVentiX to fill the server gap and give them time to move apps over on their schedule rather than a mass-migration strategy.
- MSIX achieves a higher level of Application Compatibility than App-V

No matter how you deal with them, Apps are hard. Each one is unique. Most are filled with features for consumers that you don't want in your environment. Supply chains have become active vectors for getting malware into organizations. And app lifecycles are shrinking forcing you to touch them more often. No organization can continue the old "install and forget" mindset.

MSIX offers additional safety for these application assets. Eliminating the vendor initiated updates not only makes your production environment more stable/efficient, it also allows you IT to provide the surety that the applications running are the versions that are supposed to be.

If your organization was put off by MSIX in years past, re-read the last bullet item. It is time to take another look at what it does today. .

> *PS: If your staff needs some tooling, training, or a jump-start on a Migration project, we are here at TMurgent to help on that to.*

That's the highlights, the full report follows.

1 Support by Software Vendors to release in MSIX format

Microsoft

Microsoft has continued to expand on their commitment to move their own software over to MSIX.

Just drop into a PowerShell window on your PC and type “get-appxpackage”. Much of what you will see is the background pieces needed in the OS, *framework* packages that act as layers used by other MSIX Packages.

But you’ll also see a number of other Microsoft apps that you probably use every day. Outlook (new), Teams, Notepad, and even the replacement for the command prompt, Windows Terminal. Sure, Microsoft struggled initially with the MSIX Teams app initially, but that was mostly because the customers were using older versions of the OS.

The other thing you might not notice is that when Microsoft ships you AI, it comes in MSIX for your protection. There are concerns over the possibility of bad actors polluting the models that drive AI. Microsoft has said, and we are seeing, that when they ship agents to customers, it comes in the form of a MSIX package. This ensures not only that the version you have is from Microsoft, it insures it was not tampered with enroute, during installation, or any time after installation. Those files in the package are immutable, and verified that the contents haven’t changed every time used. Of course, it also means that the update process is smooth and completely clean.

Independent 3rd-Patry Software Vendors

Microsoft’s Independent Software Vendor (ISV) Partners, who build end-user applications for the Enterprise, are releasing in MSIX format more. It might not be a tsunami, but the tide is shifting.

As predicted in earlier reports, we see this more with new applications that are written in modern coding frameworks (like .Net 7/8/9…). But some old time software is out there also.

In January we conducted a test against the over 12000 applications in the public Winget repository to determine if there were MSIX packages in the repository. In that test, we excluded anything in the Microsoft Store, just the normal Winget repository. In that test, we found over 150 packages in the repository.

The number of vendors with releases as MSIX packages in the Windows Store has also increased. We do not have any hard numbers to offer, as store entries rarely indicate if the are containerized apps or not.

The Rise of the Hybrid Installer

More surprising to us is the rise of what we are calling the “Hybrid Installer” we noticed at the end of 2025. This is a clear indication that even vendors that love their MSI and Exe based installers are finding it harder to install what they want for the end-user experience as Microsoft evolves it for their customers.

These are vendor installers that look like traditional MSI or EXE installers, but they are a combination of the traditional install along with embedded MSIX packages that they also install.

These vendors have noticed that because of changes in the operating system, especially when it comes to right-click menus, and certain shell extensions, the old way of integrating into the user session

doesn't work as well as it used to. For example, on a right click you might want the application menu item to appear on the first page rather than the user having to click "Show more options…" for what they need. They also appreciate the modern way of separating resource packs for specific OS language, region, and even personal settings like Text Scaling.

So, in addition to their normal installation, they have one or more MSIX packages in their installer that they drop down. Some appear to drop down the file and install the package as part of their installer. But others drop down the file and have code in their application that installs the MSIX package when the user runs the application. Since MSIX packages install without elevation this method works.

These hybrid installers that install as part of the installer are a challenge to repackaging efforts. Today, the only repackaging techniques to handle them would be repackage to MSI and MSIX. Both would require considerable manual manipulation, but for MSIX this is just a copy/paste between manifest files. We added detection of these MSIX packages embedded in an MSIX package in TMEditX Editor to make the packager aware.

While these hybrid installers are still under the radar today, they offer a significant challenge to someone repackaging the vendor installer. There simply isn't a way to do it other than repackaging into MSIX. That means repackaging to MSI, EXE, App-V, or any other application layering of application virtualization solution.

Even with MSIX, we do not yet have an automated way to handle this. But we do detect the presence and warn the packager. Today this means manually merging the two AppXManifest files and copying any other needed files. Fortunately the MSIX packages in the hybrid installer are quite small and simple. We will look to automate this later this year.

The Settings API

Microsoft opened up a new API for vendors to handle their application settings and personal settings. Implemented via the system.configuration.dll, it was originally for .Net applications but has been extended to also support .Net Framework and Win32 applications. We are seeing many third party vendors adopting this over traditional ini, cfg, and registry methods.

While in theory, the developer does not actually know where these settings are stored, a redirection of those files "under the covers" when running under virtualization/layering should not matter. But developers can't leave well enough along, and we have detected a few apps that use the API, but then directly manipulate the file, or assume they can use the path as a relative path to locate another file.

This was causing us some issued when putting the packages into MSIX, but thanks to the research of file system virtualization we now have a simple solution for those apps.

2 MSIX Runtime Support

The MSIX Runtime built into the operating system has seen a few improvements in both 24h2 and then 25h2. These OS versions also contained system level changes unrelated to MSIX that required us to adjust our tooling.

We see Microsoft continuing to invest in new schema extensions for future OS versions as well in both published and unpublished forms.

Most importantly, many of the past improvements are now being deployed as Windows 11 Migrations are complete and Windows Server 2025 is available for the RDS Servers. As customers migrate from Server 2022 this year, the full benefits of MSIX App Attach become reality.

While MSIX App Attach may be used in any pooled VDI (non-persistent) or RDS (semi-persistent) deployments, customers appear most keen to use it as part of an Azure Virtual Desktop (AVD) migration. While Microsoft made news last year in opening up AVD to other forms of App Attach, notably App-V, FlexApp, Omnissa, and Numecent, what fell under the radar was the changes that make App Attach better for MSIX too. Some of it was OS changes, for example supporting Shared Package Containers. Some of it was Portal. And some of it was updates to FsLogix and ease of access via Windows App, and possibly Remote App.

Microsoft did cause some confusion in their AVD Release Notes last March that stated "MSIX App Attach will be deprecated on June 1, 2025". It really meant that the term MSIX App Attach was going away in the console, replaced by the generic App Attach. The MSIX form of App Attach is definitely not going away!

Meanwhile, the format of choice for MSIX App Attach remains a question customers must grapple with. The glitches in third party tools for CimFs that occurred a while back due to undocumented changes should be a thing of the past.

I produced some performance numbers of CimFs versus VHD and VHDX in a special App Attach Report Card for 2024 a couple of years ago showing better performance for CimFx in a single-user scenario. And as I stated in the paper, the architecture of CimFs should be much better than traditional image formats in a production scale test. But I hadn't actually tested it. As the VHD/VHDX are more common formats with more available tooling, and the format of choice for most App Layering products, plus the tooling glitches I mentioned of a couple of years ago, it seems most customers have shied away from CimFs.

Microsoft (Jim Moyle) did publicly present results of scale testing at the WP Ninja (US) conference in December (I saw it live but have no link to offer). Those results validated, and possibly exceeded, my assumptions of performance at scale. Customers already using App Attach might not want to reconsider the choice, but new App Attach customers now have a choice to make.

3 Repackaging Testing

At the end of the day, we need to be able to deploy the apps. Without a lot of vendor supplied apps available, we can work the path of repackaging existing applications into MSIX and testing those. In this section, as in years past, 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 131 to improve coverage of newer apps. Most of the applications being tested 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. Some of the expanded list of applications tested include new applications built for .Net 8/9, and even a couple of local AI apps not from Microsoft.

These applications were repackaged and tested three times, all on Windows 11 25H2:

- ~~Using Microsoft App-V.~~
- Using the Microsoft MSIX Packaging Tool (2024.405 release) without the PSF. (packaged not tested)
- Using the Microsoft MSIX Packaging Tool (2024.405 release) with TMEditX Editor 7.1.0.0.

Many of the advances made for repackaging to MSIX come from improvements to the TMEditX Editor. Although much of the focus on that tool was to make it easier to achieve good results, technical fixups were added that made longstanding apps now packageable.

The primary areas addressed with new technical fixes in the editor include:

- Com/Active-X. We should now have pretty much full support.
- Shortcuts. Software Vendors do the strangest things. We should now be able to handle most everything the vendor install throws our way.
- Child process launching. You remember what I just said about Software Vendors, right? In particular vendors that are used to writing for Linux or Mac and released cross-platform versions for Windows do strange things too. While there is more work to do here, we picked up a some of those apps.
- Flexible Virtualization.
- The ability to support apps that write to HKLM. The solution was borrowed from App-V where we redirect it to a special place in the virtual HKCU area for the package.
- Windows Firewall Rules. Love 'em or hate 'em from a vendor package, we have you covered now. See research paper: Windows Firewall Rules and MSIX Packages
- Although not part of the apps we tested for this report, we added support to recognize items in vendor supplied MSIX packages and customize those also. Think things like autoupdate extensions that you'll want to remove.

Unlike previous years, we did not test with App-V packages. There has been no work done by Microsoft on App-V, so we pretty much know that last year's results would be the same for the same set of apps. As for the common apps in the list that we upgraded this year to new versions, and especially the new apps added to the list, we know from experience that this would have dropped the App-V number down as these new .Net8/9 apps and Hybrid Installers are problematic for App-V (as I'm sure they are for the third party vendor formats).

We also did not test the raw packages from the Microsoft MSIX Packaging Tool (MMPT) with their own PSF version, nor the MSIX Packaging Tool with PsfTooling this year, but anecdotally we can suggest that we would expect:

- Using Microsoft's MMPT without PSF we would probably see about 33% compatibility this year.
- Using Microsoft's PSF version we would probably see around 50% compatibility maximum, after much manual effort.
- Using PsfTooling to add the PSF in the MSIX Packaging Tool would provide about 60-65% compatibility.

You'll have to read on to see the test results using the MMPT plus TMEditX Editor are this year.

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 "subjective" buckets). But we must 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.

Additionally, some of the result graphics offers an additional summarized version of the results in bar form that combines the first five categories into a failed status. This creates a simpler subjective view where the reader can apply their own subjective view of compatibility with categories of “Good”, “Maybe”, “Maybe not”, and “No way”.

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.

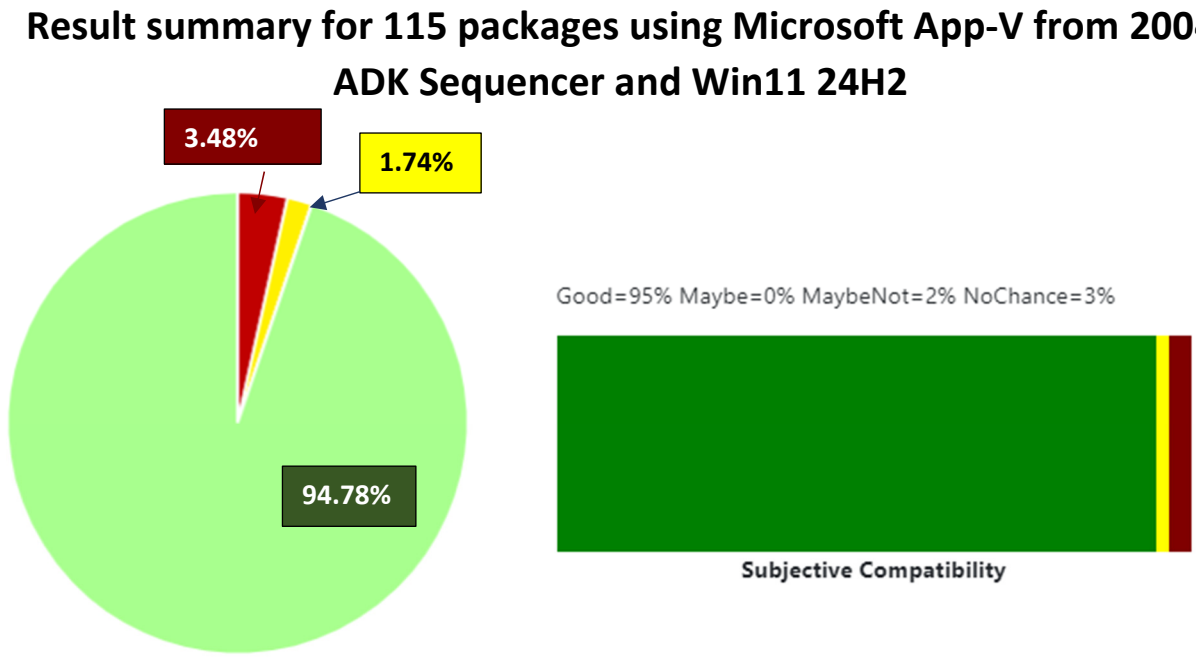
The applications chosen for testing come from a combination of large and small commercial applications we see large enterprises deploying in our practice, plus a number of “special purpose” applications we wrote to mimic a specific pattern that developers might use (for example, an app with 40 ways to set up a file type association). Applications that clearly are not suitable for app virtualization and layering, such as drivers and traditional office plugins were not considered for testing.

3.1 Comparative results using Microsoft App-V

While we did not test App-V this year, here is the closest result from the 2024 report card.

The 115 packages were packaged on Windows 11 24H2 and tested on the same OS. The 2004 ADK Sequencer was used and packages were further fixed up by TMEdit which improved the results

.



Not surprisingly, these results were very good using seasoned tooling and collective wisdom on techniques to package with App-V. Of note is that some of the newer applications that use the newer DotNet runtimes had issues. As vendors release more of these apps, App-V numbers may slide further from the traditional “98%” number.

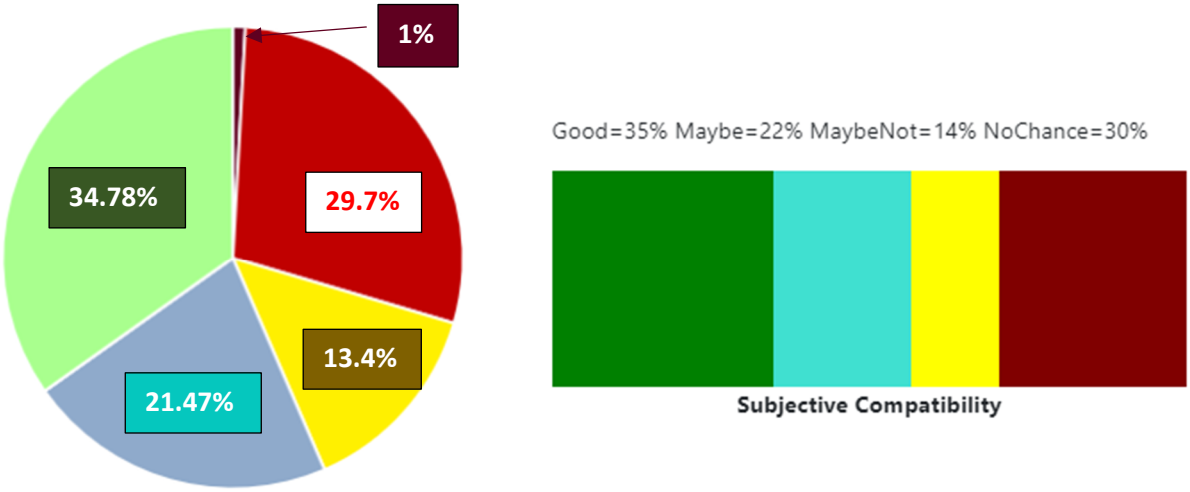
We know that the number would have dropped a little from this if we had run the new application list against App-V.

3.2 Results from Packaging with Microsoft MSIX Packaging Tool

While we did package the full list this year, with no changes to the MMPT, we expect similar results to that of last year's test, shown here.

The same 115 applications were packaging using the 2024.405 release of the MMPT, without trying to add the older version of the PSF or other manual editing. These tests use current best practices for packaging, without the use of additional tooling or extraordinary measures such as manual manifest editing.

Result summary for 115 packages using MMPT NoPsf (24H2)



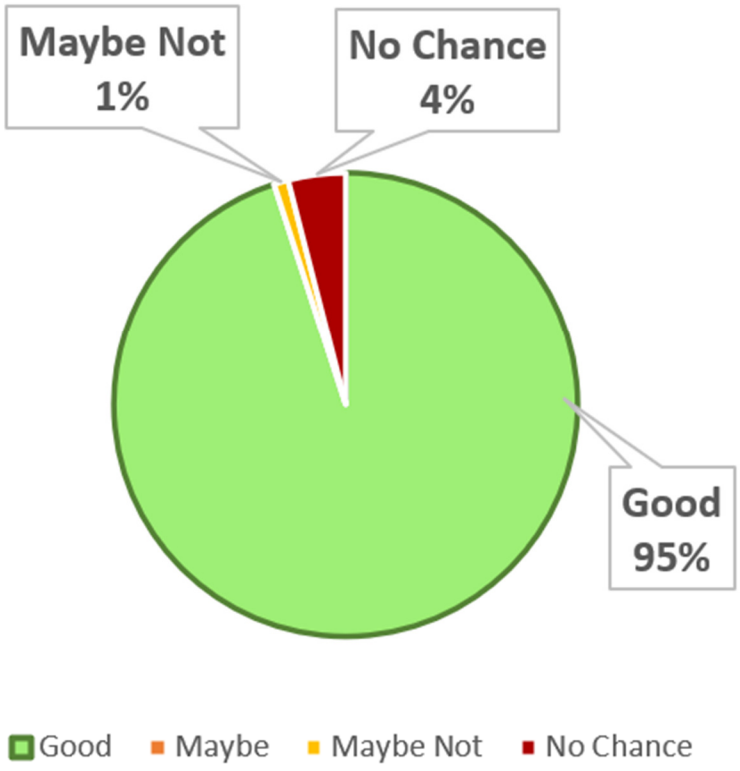
3.3 Results from Packaging with Microsoft MSIX Packaging Tool plus TMEditX

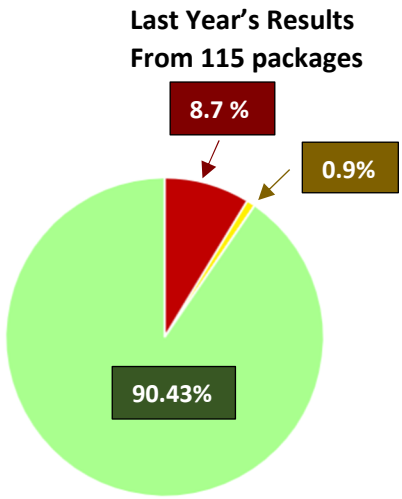
The packaging for these tests were fully automated.

- Vendor installers were wrapped to remove updaters and perform typical customizations using our free PowerShell module github.com/TimMangan/PassiveInstall.
- The MMPT was used for the initial capture using remote capture VMs. The PowerShell framework for that automation is available in the TimMangan github repository as msix-packaging2. It is an enhancement to the original Microsoft framework.
- The packages were post-processed by the same framework to enhance and fixup the packages by using TMEditX Editor 7.1.0.0 to inject and configure the Package Support Framework (PSF) into the package as well as make additional improvements available in MSIX but not through the MMPT. (TMEditX Editor is a licensed product and information on TMEditX is available https://www.tmurgent.com/AppV/en/buy/tmeditx/tmeditx-about) . This version of TMEditX Editor uses version 2026.02.02 from our fork of the PSF hosted on GitHub here: TimMangan/MSIX-PackageSupportFramework

Click one button, get 131 packages. Then test them…

Result summary for 131 packages using MMPT/TMEditX (25H2)

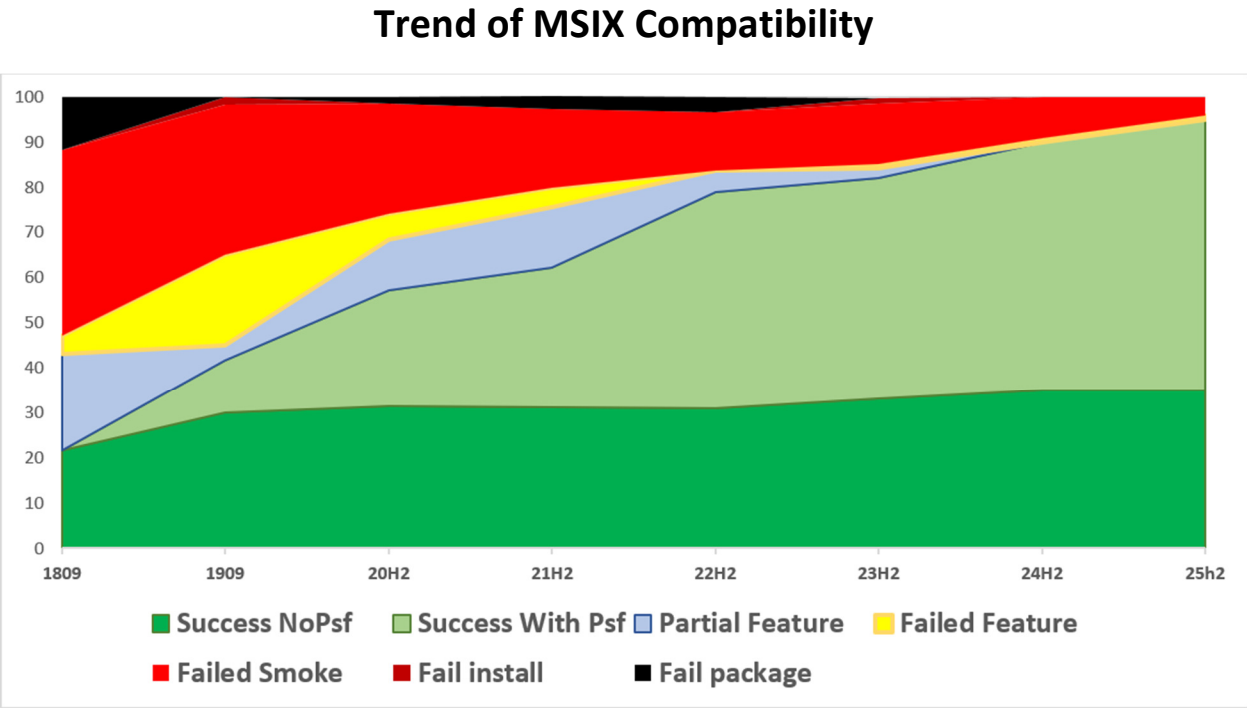




Even with the more difficult application mix, the high-fidelity result managed to pop above the 95% mark! As seen by comparing versus the results from just last year, we are continuing to make great progress.

3.4 The trend

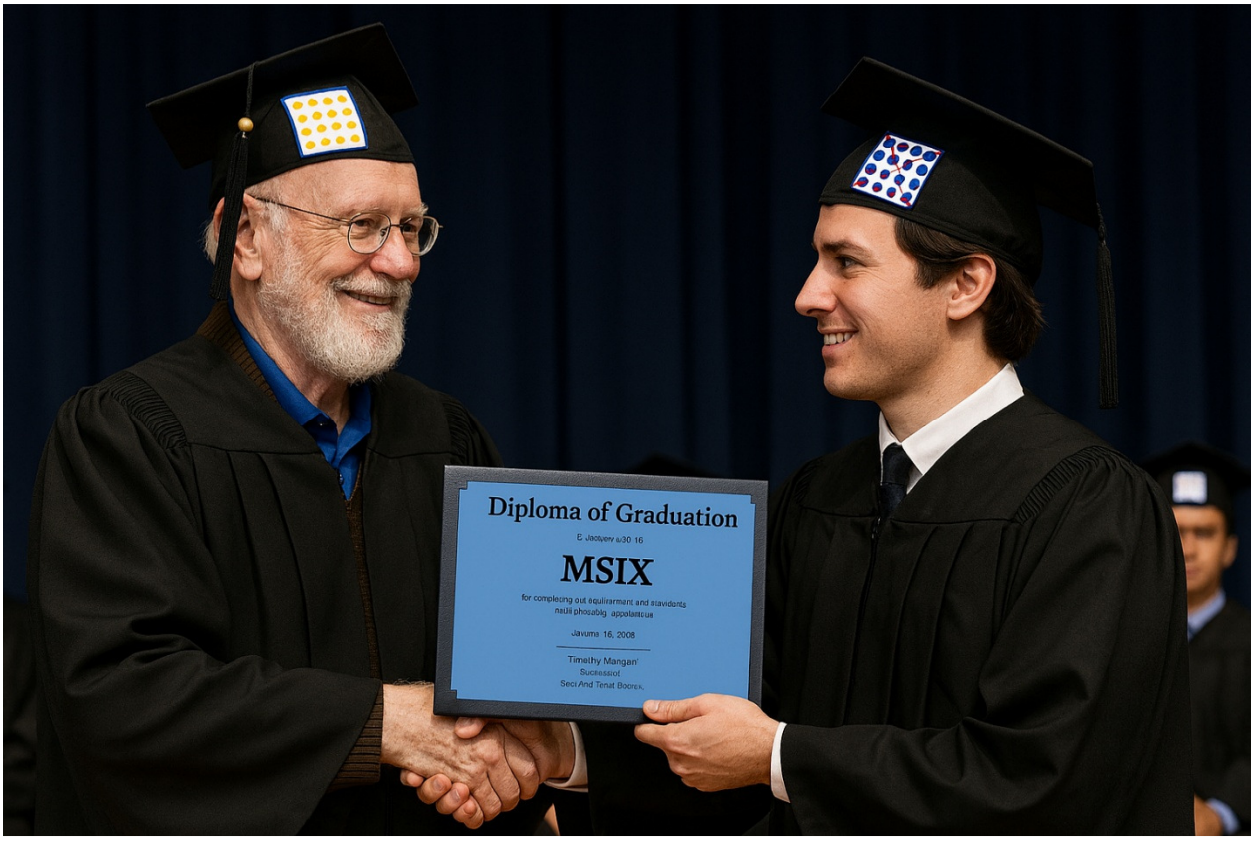
Now that we are in the *eighth* year of this testing, we can look at the numbers to see how the trend has been going.



A year ago we said that MSIX was ready for the Enterprise. This year, it is better and easier to take on. I honestly did not expect that MSIX would surpass App-V for another year or so. I have been surprised at the improvements for this year.

3.5 Future Work

It is time to turn the page on MSIX not being ready. This may be the final Report Card; I think we've reached the point to kill that meme. But work will continue, and I'll likely report back with updates in another form. **MSIX has graduated!**



While Microsoft rarely, if ever, publicly discusses there future plans for MSIX, I can talk about my plans to improve MSIX packaging for traditional apps. Typically I have a full list ready by the time this report comes out, however, I haven't had time to work that out yet. But this is what I am thinking about.

Five areas of challenges that I have identified for this year are for the support of additional types of packages have been identified outside directly from vendors, and remaining types of apps that can be problematic:

1. Additional blockers preventing organizations from adopting MSIX. Outside of specific apps, what are your blockers?
2. Improved support UTF8 apps. We see this mostly in Linux apps ported to Windows.
3. Improved support for mis-behaving child processes.
4. Work-flow for modification packages.
5. Guidance for Developers working within an Enterprise, to help them make their in-house apps more compatible for 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 19 years, and by Citrix for about 15 as a CTP/Fellow (before they dropped the program), Tim is generally known as "The Big Kahuna", having led the effort to build the original version of App-V at Softricity and builds many free and paid-for tools for packaging under both App-V and now MSIX.

TMurgent provides consulting and training around application and desktop deployments. Our "*Packaging for MSIX*" training classes are sought after by IT Professional Desktop Engineers around the world, with both in-person and remote-live training at training centers in US and Europe, or private on-site or remote classes upon request.

TMurgent Technologies, LLP is an independent company engaged in the application 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, including TMEditX, the software used to produce the better results shown in this report testing.

Those interested in MSIX are encouraged to download one of our two community eBooks on MSIX. Each is over 200 pages and co-authored with leading community technologists for a thorough coverage:

- **For the IT Pro:** MSIX Packaging Fundamentals (tmurgent.com)
- **For the Developer:** A Developer's Guide To MSIX (tmurgent.com)

Our other main book of interest is **The Application Book** **The Application Book,** which is for anyone packaging up, or trouble-shooting, applications for any format.

As an independent contractor, TMurgent 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 currently.

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. The Vendors have not been given an opportunity to review or correct potential factual errors in this report prior to publishing, however they may contact me if they feel there are errors in the report that should be addressed.