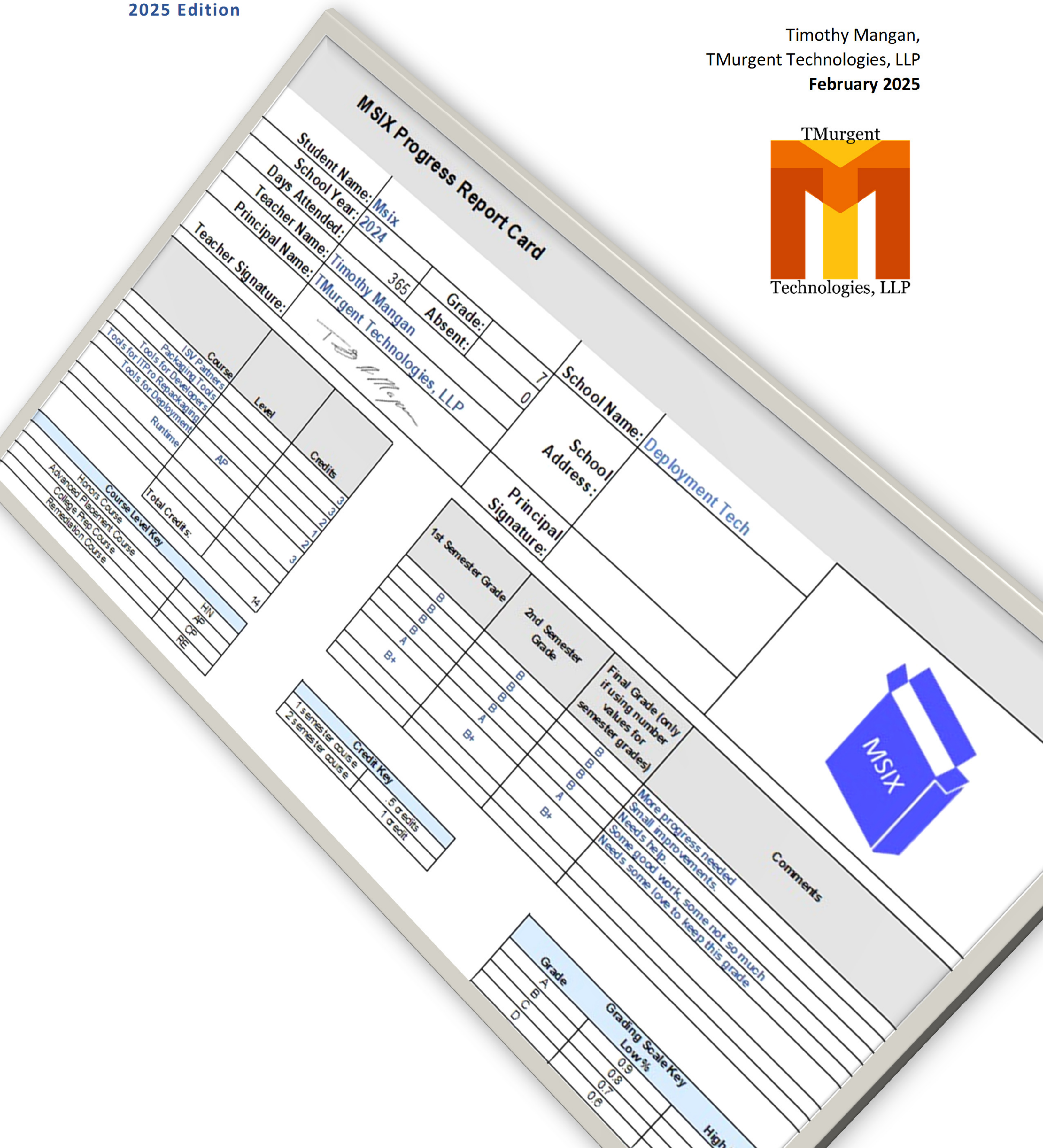
The MSIX Report Card

2025 Edition

Timothy Mangan,
TMurgent Technologies, LLP
February 2025



MSIX Progress Report Card

Student Name:	Msix		
School Year:	2024		
Days Attended:	365	Grade:	7
Teacher Name:	Timothy Mangan	Absent:	0
Principal Name:	TMurgent Technologies, LLP		
Teacher Signature:	[illegible]		
School Name:	Deployment Tech		
School Address:			
Principal Signature:			

Course	Level	Credits	1st Semester Grade	2nd Semester Grade	Final Grade (only if using number values for semester grades)	Comments
ISV Partners		3	B	B	B	More progress needed
Packaging Tools		3	B	B	B	Small improvements.
Tools for Developers		2	B	B	B	Needs help
Tools for ITPro Repackaging		1	B	B	B	Some good work, some not so much
Tools for Deployment	AP	2	A	A	A	Needs some love to keep this grade
Runtime		3	B+	B+	B+	
Total Credits:		14				

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	

Introduction

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. We are still on that road and are likely to be for some time. In recent years customers have become outspoken about the issues they encounter and question just how committed Microsoft is to this path.

This 2025 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 and
- http://m6conf.com/index.php/reportcard/46-report-card-2024

We can summarize the changes this year as follows:

- Customers are completing their hardware refresh cycles and adopting Windows 11, which enables some of the important additions that Microsoft added to MSIX to support different kinds of applications.
- Microsoft has continued to increase support for MSIX based applications in their own offerings. There are many smaller apps, but also Edge, Teams, and the new Outlook are now MSIX applications with significant functionality. There remains much to be done with their own apps, but we are seeing progress.
- Third party vendors have also made progress in moving to MSIX, and we are glad to see the increased offerings with new applications written against the newer .Net versions. On that front Microsoft has made it easier with tooling for developers to convert existing applications to the new developer platforms, but most older applications still have challenges that Microsoft doesn’t provide enough help for. These are mostly the same challenges we have learned to overcome in repackaging to MSIX, but the vendors are left without a guide.

- The level of application compatibility that can be achieved with repackaging existing applications into MSIX by IT Professionals continued to improve this year. We believe that level is sufficient for an enterprise to choose MSIX as their first choice for repackaging.
- While most enterprises say they want a choice that can be their only method to repackage, that choice does not exist and enterprises must become content with needing a primary and backup choice. Now that Microsoft has withdrawn the deprecation of Microsoft App-V, customers have more time on that as choice for the backup plan.
- Our own efforts this year have been focused using TMEditX tooling to augment Microsoft's MSIX Packaging Tool. In most cases, this work can be automated, however some applications require manual efforts
- MSIX App Attach is continuing to be a delivery vehicle for certain scenarios, with Vendors like VMWare, Citrix, and others having added support. At the end of the year, Microsoft opened up AppAttach on AVD to support other packaging formats, such as App-V and other 3rd party vendors. It will be interesting to see what Enterprises do.

My intent is to update this report card in (roughly) 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 Vendors to release in MSIX format

B

Microsoft’s Independent Software Vendor (ISV) Partners, who build end-user applications for the Enterprise, stay at a “B” for releasing software products in an MSIX format.

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

For direct delivery from the vendor, we are seeing apps appear that are built using .Net 6,7,8, and now 9. Applications built using these runtimes are inherently compatible with running inside the MSIX container. However, Microsoft opened up the ability for the vendors to release in an “unpackaged” form as well. So we are seeing a mix of how the vendors release and this impacts how Enterprise customers have to process the applications for internal distribution.

2 Support by Tooling Vendors

B

Tooling Vendors dropped from “B+” to “B” this year.

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.

2.1 Tooling Vendor Support for Developers

B

Microsoft has been doing a lot for developers, not only in improving the SDK, but advanced in Visual Studio itself. The project converter to convert to the newer .Net is fantastic.

The SDK for developers has matured and supports most of APIs that developers require. Much more is to be done, but Microsoft has been delivering on their promises.

Visual Studio improvements have been outstanding. The new GitHub Copilot integration is just fantastic and will turbocharge software development across the board.

2.2 Tooling Vendor Support for Repackaging

B

Software Vendors supporting IT Pro Repackaging of applications into MSIX earned a “B” this year, down from “B+” last year. These established vendors on the packaging side were very active in the MSIX community from the beginning, but we hear concerns and frustration from them regarding Microsoft’s level of commitment.

Although Microsoft continues to say that they are investing in their MSIX Packaging tool, we have seen very little lately.

TMurgent’s tools, PsfTooling and especially TMEditX, offer an improved PSF version (and ease of implementation), and an ever increasing set of remediations beyond the PSF.

Meanwhile, the Package Support Framework (PSF) in the TimMangan fork has advanced the game with the usual improvements. The biggest change this year was the introduction of a new component PsfFtaCom. The component is part of a comprehensive shift in how file type associations, shell extensions, and COM components are declared in the AppXManifest. This overcomes limitations around uniqueness and completeness rules that exist in the Schema sets, while working within what Microsoft defined.]

Other packaging vendors also include the PSF in their own repackaging products, combining the equivalent of PsfTooling and the MMPT into a single experience. Some appear to be following my fork of the PSF to improve their products. Seemingly all are doing more to make it easier for the people doing the repackaging to be successful, and all in quite different ways.

The growth in this space this year has been in vendors that have offerings to help IT with more of the application lifecycle in one tool. They often include their own packaging software, or repackage/refer to other packaging tools in their workflow. While these vendors are not exclusive to a particular packaging format, MSIX is driving their businesses. Several of them now include integration with TMEditX.

2.3 Tooling for Distribution

A

Software Vendors with support for MSIX Distribution at a solid "A" again this year.

There is not a lot that is new to talk about here this year.

3 MSIX Runtime Support

B+

The MSIX Runtime remains at a "B+" grade this year. On one hand, we haven't seen a lot that is new this year. But the other hand, some of the work they did previously that we could not use is now coming on line with customers adopting Windows 11.

We do see signs of Microsoft continuing to expand upon the capabilities of MSIX with new schema definitions, however it will not be until next year (at least) before the OS support for the new extensions becomes available.

4 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 115 to improve coverage. Most 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. Some of the expanded list of applications tested include new applications built for .Net 7/8/9. 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.   The applications that were One new app in the list was an new photo app that utilizes local AI to enhance pictures using the GPU (or CPU as fallback). That this app works so well was a pleasant surprise.

These applications were repackaged and tested three times, all on Windows 11 23H2 and 24H4:

- Using Microsoft App-V.
- Using the Microsoft MSIX Packaging Tool (2024.405 release) without the PSF.
- Using the Microsoft MSIX Packaging Tool (2024.405 release) with TMEditX 5.5.0.0.

We did not test on Windows 10 this year. Many of the advances made for repackaging depend upon MSIX improvements that Microsoft made available only for Windows 11. With the end of life for Windows 10 looming, we felt that Windows 10 testing would not be relevant. Although we tested on both 23H2 and 24H2, the test results reported are for 23H2. Unusually, a few applications had issues when run on the newer OS, but we believe these to be OS issues that Microsoft should be addressing in the monthly updates.

We also did not test using the Microsoft MSIX Packaging Tool with their own PSF version, nor the MSIX Packaging Tool with PsfTooling, but anecdotally we can suggest that we would expect:

- Using Microsoft's PSF version we would probably see around a 50% compatibility maximum, after much manual effort.
- Using PsfTooling to add the PSF in the MSIX Packaging Tool would provide about 60-65% compatibility.

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, this year 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”.

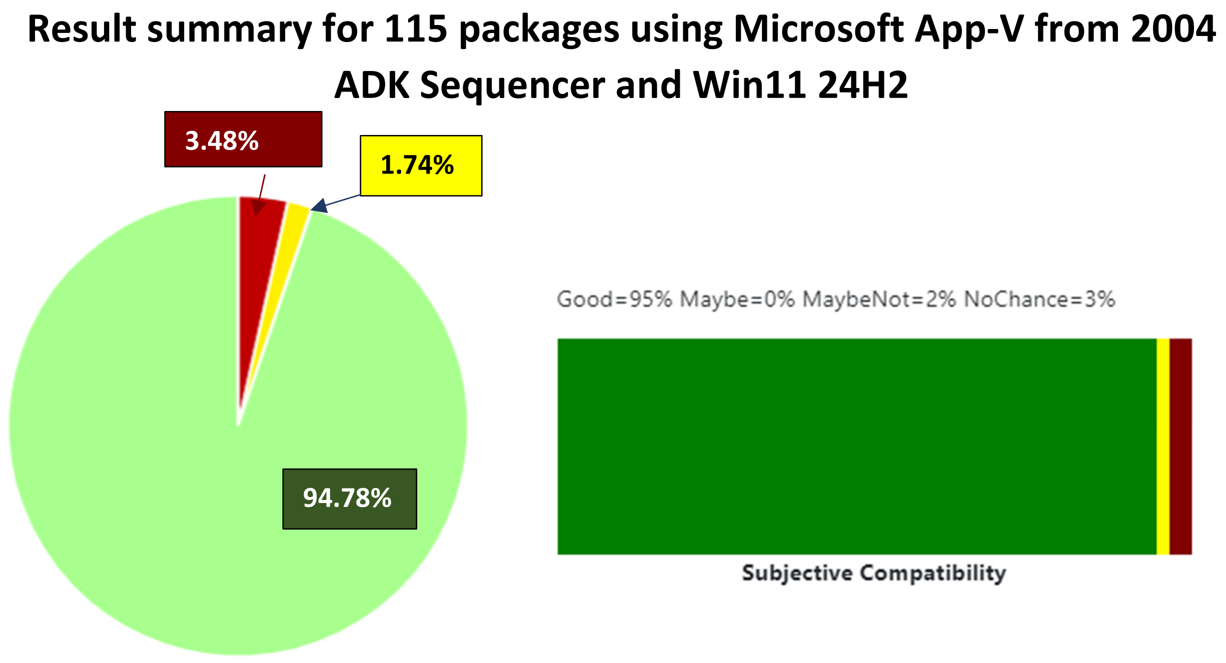
Finally, it is worth noting that the testing performed was using recapturing techniques. There may be ways to build a MSIX package using traditional install builder tools from these same vendors that have been extended to support MSIX, however that was outside the scope of this testing.

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 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

.

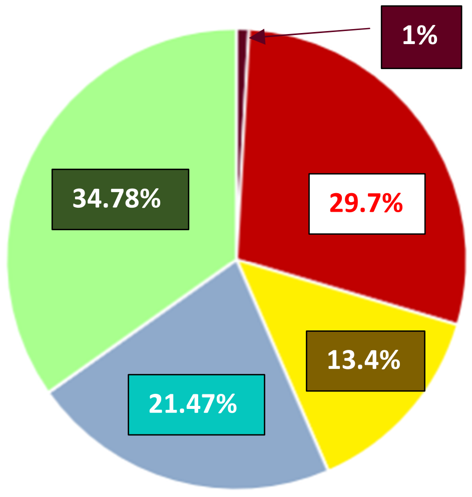


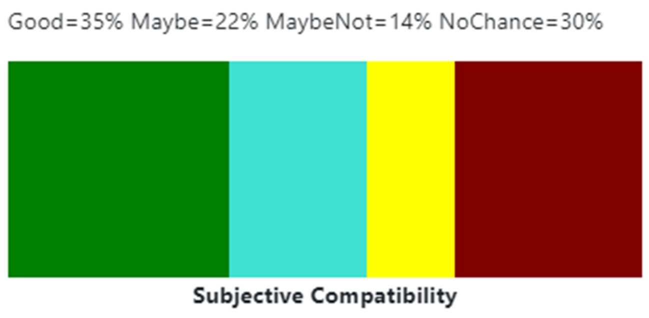
Not surprisingly, these results were very good using seasoned tooling and collective wisdom on techniques to package with App-V. Of note is that some of the newer applications that use the newer DotNet runtimes had issues. As vendors release more of these apps, App-V numbers may slide further from the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

4.2 Results from Packaging with Microsoft MSIX Packaging Tool

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)





Although the numbers changed slightly, I consider the results to be about the same as per last year's results for the same unchanged apps.

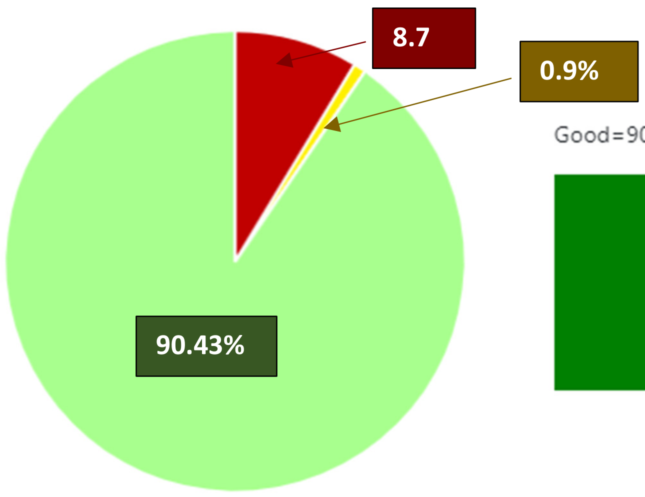
Except for a few cases, these packages were fully automated using remote package capture techniques. The reason that a few of the apps were manually packaged was that the disabled Windows Updates were re-enabled by the OS during packaging causing bad packages, or installers that refused to automatically install.

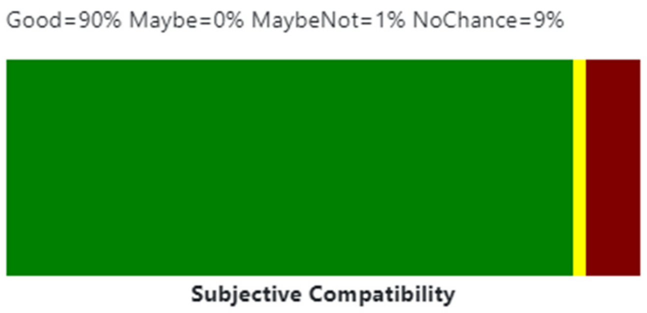
4.3 Results from Packaging with Microsoft MSIX Packaging Tool and with PSF

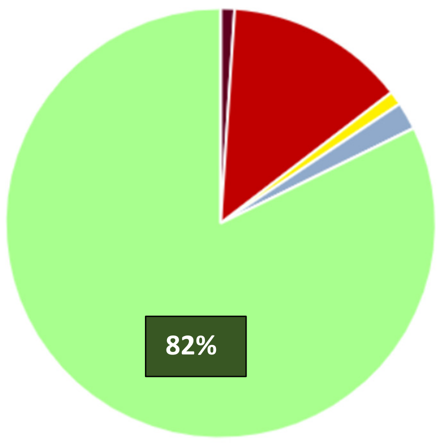
For these tests, the MMPT version 2023.1212 was also used, but the packaging process was enhanced by using TMEditX 4.0.0.7 to inject and configure the Package Support Framework (PSF) into the package as well as make additional improvements available in MSIX but not through the MMPT. (Information on TMEditX is available https://www.tmurgent.com/AppV/en/buy/tmeditx/tmeditx-about) .

For the most part, TMEditX was used via automation as well. When analysis indicated the need, the new MfrFixup (not available in the Microsoft PSF fork) in ILV-Aware mode and InstalledLocationVirtualization, the DynamicLibrayFixup, and EnvVarFixup, were added. In general, the RegLegacyFixup was automatically added to any package that analysis showed that the PSF was needed. This version of the PSF used contains a build of the latest PSF source code from GitHub available as of the beginning of January 2024 (v.2024.01.02). The same set of 96 Apps were tested.

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







Last Year's Result

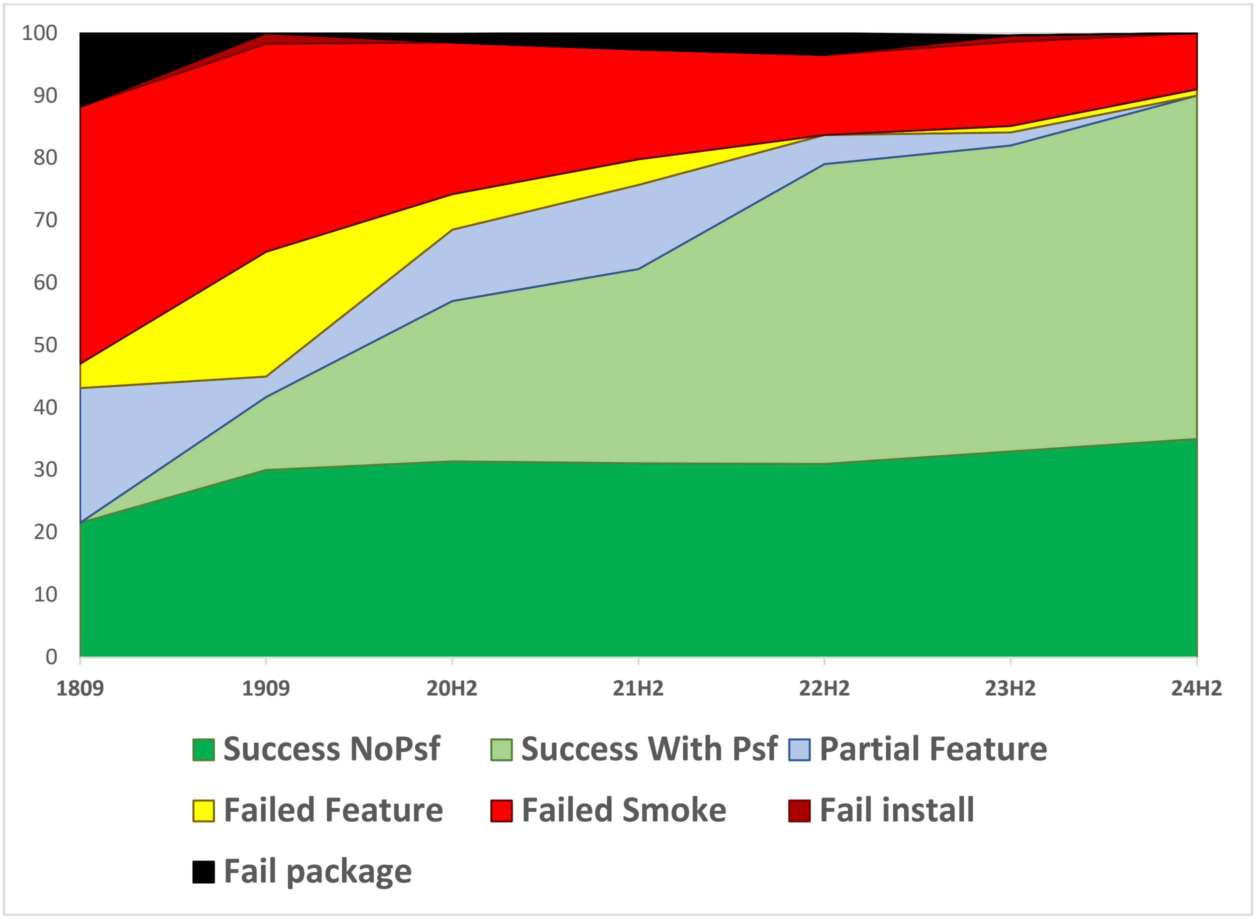
Even with the more difficult application mix, the high-fidelity result managed to pop above the 90% mark! As seen by comparing versus the results from just last year, we are continuing to make great progress.

But more striking than improved possibilities was the reduction in amount of effort that it took to produce those packages due to improvements in the analysis and available fixes. Using just the automated fixes alone allowed about 70% of the apps to achieve high-fidelity compatibility this year.

4.4 The trend

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

Trend of MSIX Compatibility



Improvement in application compatibility over the previous year was significant this time. Most of that is due to the PsfFtaCom work, especially in the area of COM support. There remains additional COM scenarios to be addressed, and we hope to make more progress on that this year.

As noted earlier, we are already beginning to see issues with some newer apps. Although MSIX is doing better than App-V at these in general, we likely will have to do more work to determine how to get some of these new apps working as well.

4.5 Future Work

While Microsoft rarely, if ever, publicly discusses there future plans for MSIX, I can talk about my plans to improve the Application Compatibility for traditional apps repackaged MSIX packages.

Three areas of challenges for the support of additional types of packages have been identified and attempts to address them are under way for 2025:

1. Improved support for COM. PsfFtaCom allowed us to register the vast majority of COM components that were being ignored by the MMPT. In one apps, we have over 700 unregistered components and now have only a half-dozen. Whereas before it just looked like a forest, we now have the ability to see species of the trees and attack them this year as a sub-group.
2. Some work was done at the end of the year to better recognize and categorize package dependencies. Work on doing something about that is still to come.
3. We are seeing apps with different kinds of relationships with the newer DotNet Runtimes. We need to be able to analyze what they need and find solutions for those with issues.
4. The PSF needs more work as well. There are alternative APIs that applications sometimes use for which the PSF has no support. While these are quite rare, progress on the overall compatibility number requires more work here.

There is no certainty that changes to solve these issues will solve all of the problems of any specific application (the tough ones often have additional issues hidden by the one you see at first), I have hope to see measurable improvements in the numbers for this report next year.

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 18 years, and by Citrix for about 15 as a CTP/Fellow, 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.

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.