happiest minds
The Mindful IT Company
Born **Digital** . Born **Agile**

Next Generation Application Delivery Model - MSIX

Table of CONTENTS

- Introduction
- New model of pacakging
- Advantages of MSIX
- Architecture of an MSIX package
- MSIX packaging tool
- Package support framwork
- Distribute your MSIX package
- Conclusion
- Author Bio

Introduction

Digital innovation and technology have made us rethink the way we live today. All most everything surrounding us has become intelligent and innovative in terms of its functionality, features or look. Organizations have accelerated their journey to digitization by many folds to meet the demands of the new-age customers. However, the sad part is that many organizations still rely on age-old traditional applications because it is challenging to replace them with modern technology Eventually, securing these old technologies requires extra effort or becomes very challenging for the security team to protect against today's advanced threats.

INTRODUCTION OF MSIX

MSIX was first introduced in 2018 October with a vision to replace all existing package delivery models.

During Windows 8 launch, Microsoft introduced a new kind of application model called Windows Store apps, based on the new framework called Windows Runtime to help developers access and develop code. Unfortunately, it made it more difficult for the developer to compile and failed to compete with the early application delivery model.

During Microsoft 10 launch, they introduced a Universal Platform as the successor of the Windows Runtime. This Universal Windows Platform provided a consistent set of APIs to access all the functionalities exposed by the core network, storage, and sensors.

Microsoft announced a new packaging format called MSIX, 2018, in October. The aim is to replace all existing package delivery models.

FEATURES OF MSIX

- Microsoft Store deployments
- Containerized applications (including win32 executables)
- Reliable installation (99% as Microsoft states)
- Network optimization through downloading only the 64k block
- Easy integration within infrastructure management tools like SCCM or Intune

EVOLUTION OF MSIX:

Windows 8 first introduced the called AppX. This model was not flexible enough compared to its previous MSI/ APP-V, as it only supported Windows Store Applications.
The main drawback of this application was that it allowed package deployment only from the store and enabled sideloading only for development purposes or on special licenses.
During the launch of Windows 10, further enhancement was made with the above packaging format using more suitable application deployment methods.

- The initial release of **Windows 10** removed the requirement to have a special license to sideload enterprise applications packaged using AppX

- **Version 1511** - Turned on default option to allow package sideloading without changing the windows configuration, enabled to install package coming from the trusted source

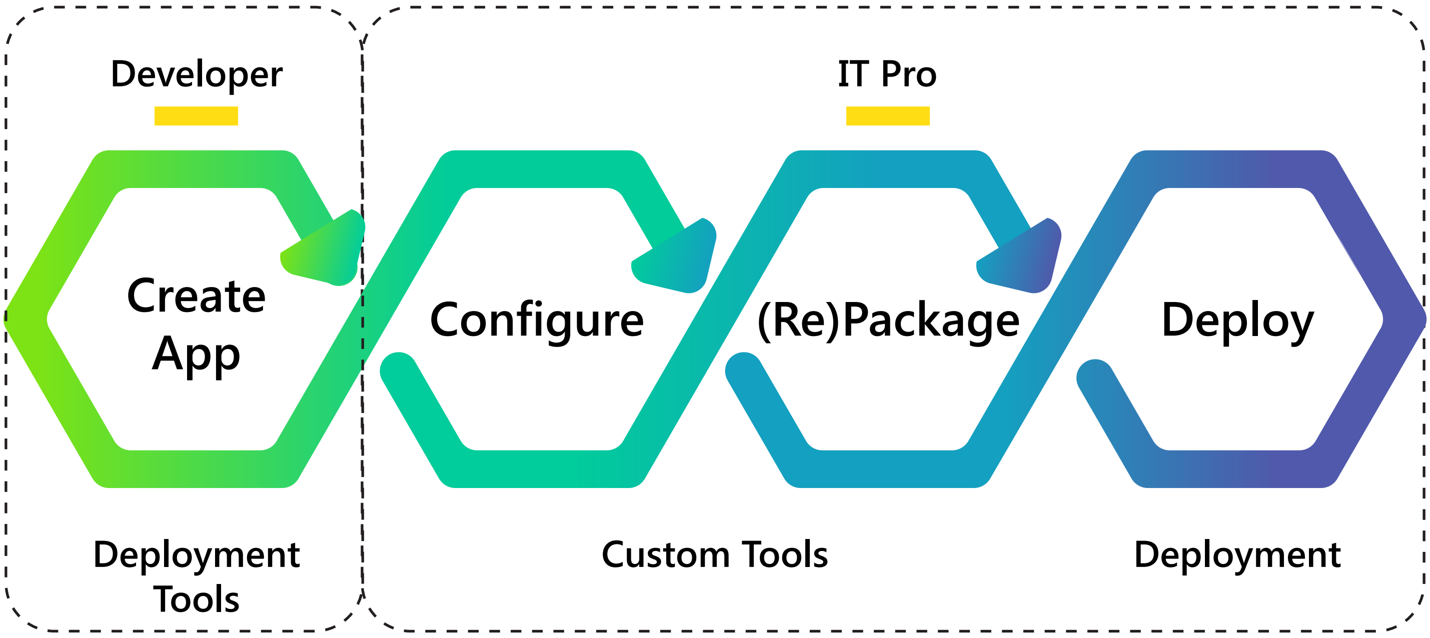
- **Version 1607** - The Anniversary update introduced the Desktop Bridge, extending the AppX packaging format to the Win32 ecosystem. This different application-built format took advantage of the new deployment format

- **Version 1809** - The Introduction of MSIX built on the AppX fundamentals

- **MSIX** - It is open-source and cross-platform support. It also supports significant packaging formats such as mobile (Android and IOS) and Desktop (Windows, macOS, Linux)

Advantages OF MSIX

Deployment is always a difficult job not only for the developer but also for IT Professionals. In most cases, IT professionals need to rely on the deployment process. They need to customize the installer to accommodate the requirements of the company.
Microsoft had created the custom tools for MSI and App-V, which allowed users to take the installer provided by the developer and customize the development process.



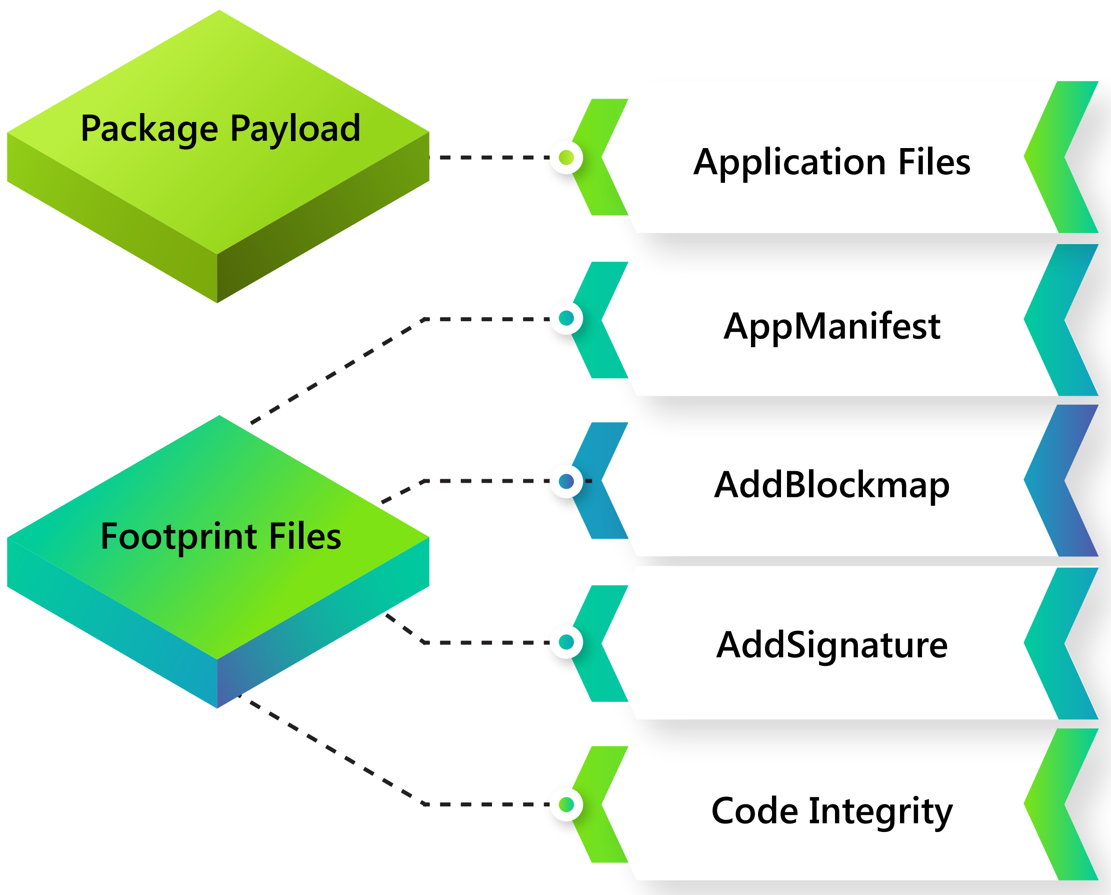
The above figure exhibits the typical application lifecycle in the IT pro world. MSIX helps IT pros independently deploying Windows updates, application updates, or customizations. Thus, reducing the repackaging application efforts. MSIX provides a reliable install boasting and to have an excellent success rate.

- Package and distribute content beyond Windows (extensible to iOS, Android, macOS & Linux)
- Optimizes disk space (leveraging single instance storage across apps and users)
- Optimizes network usage (streaming differential updates at the block level)
- OS Managed installations (much of the logic and interface of software installation will now be handled by the operating system)
- Windows provides integrity for applications (Tamper protection, policy controls)

Architecture

OF AN MSIX PACKAGE

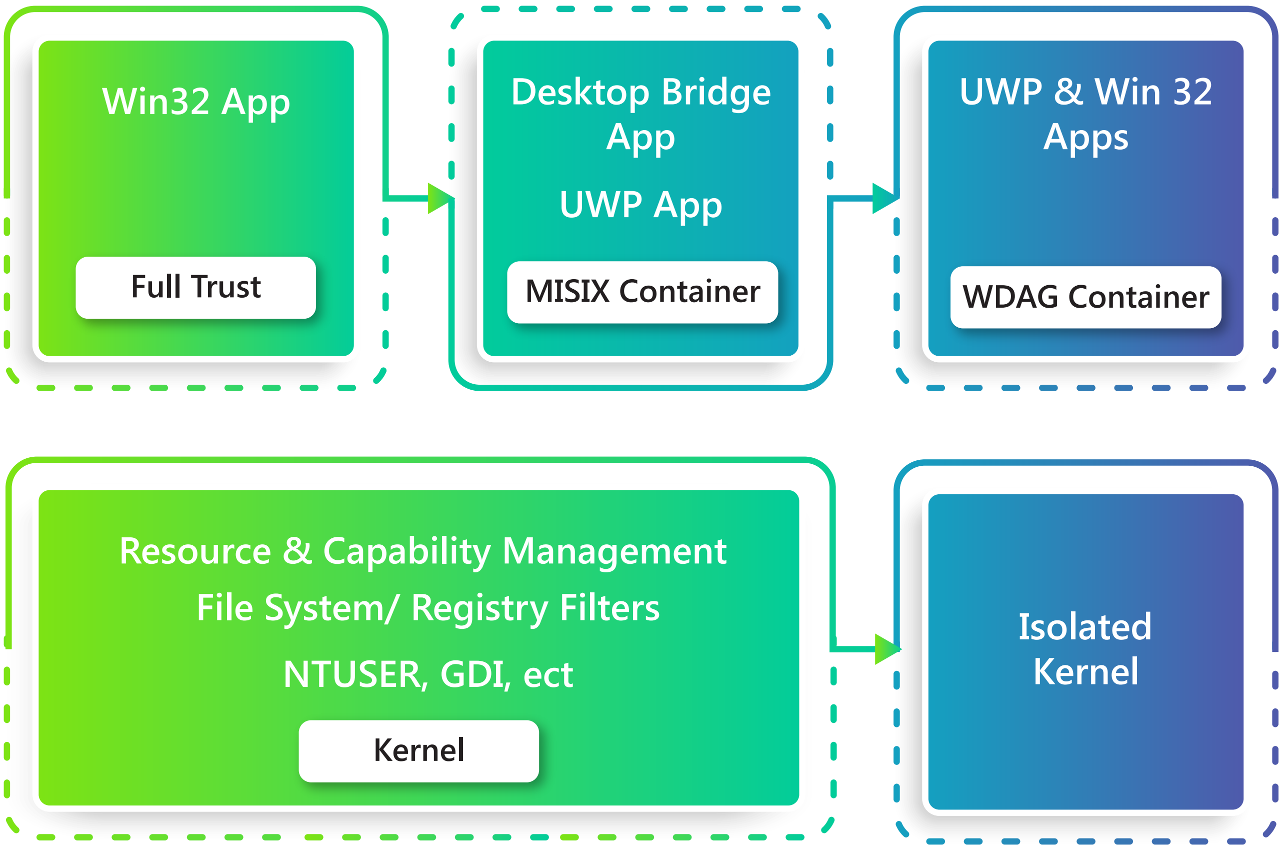
An msix package is a simple file container for an application. Inside it, you will find all the files that make up your application, plus a set of special files and folders:



AppManifest.xml	- The manifest file, which describes the application.
VFS	- A special folder used to abstract the file system and provide virtualized access to system folders.
Assets	- The folder that contains the default image assets leveraged by Windows 10. These are the icons used for the Start menu entry, the tile, and the icon in the taskbar.
A set of files with the .dat extension	- These files map the various registry hives that are part of the system registry, and they include the keys that need to be created during the deployment phase.

MSIX CONTAINER

Container support from desktop bridge apps with one UWP apps



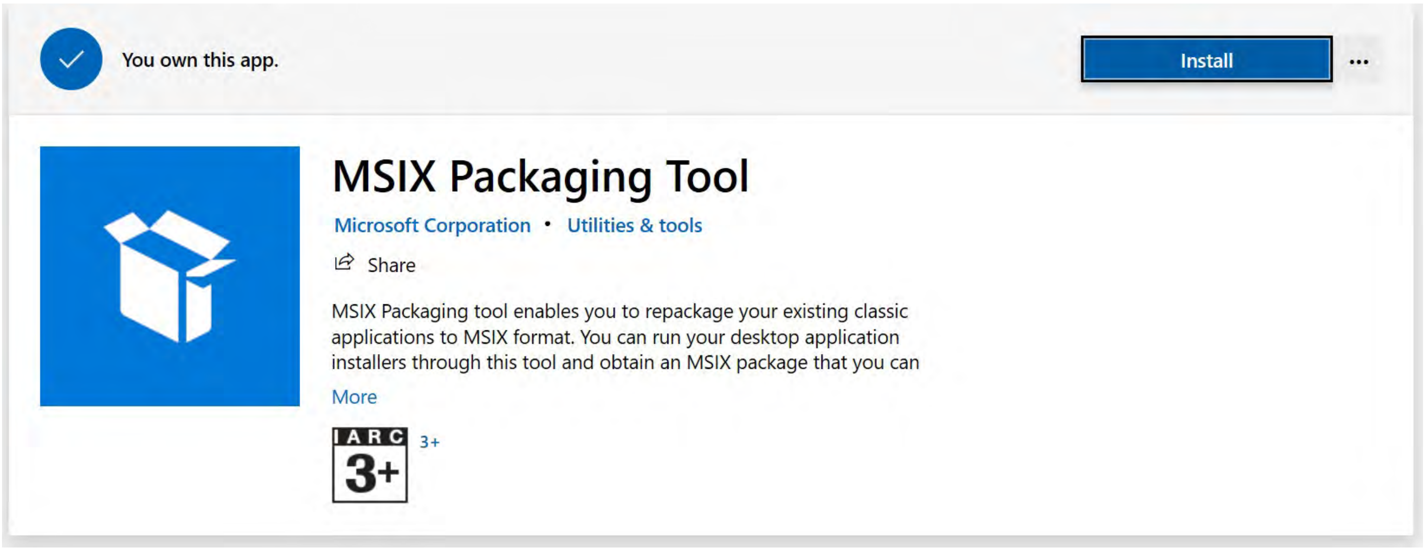
Win32 applications packaged as MSIX will run only on windows 10 S mode, where win 32 apps run on ARM devices with emulation support.

Packaging Tool

MSIX

To package the application in a new format (MSIX), Microsoft has released a new free tool called MSIX packaging tool. This tool uses an installer from a self-executable application rather than source code. The minimum requirement of the tool to use is Windows 10, version 1809 which is handled by a special driver that comes preinstalled with Windows 10. The tool will monitor all the changes performed by the installer to the operating system, from the deployment of files tothe creation of registry keys. All these changes will be saved and then included inside the final MSIX package generated by the tool. This tool is smart enough to handle all the exceptional cases automatically.

Ex: if the installer copies one or more files inside one of the system folders, the tool will automatically copy them in the proper


You own this app.

MSIX Packaging Tool

Microsoft Corporation • Utilities & tools

MSIX Packaging tool enables you to repackage your existing classic applications to MSIX format. You can run your desktop application installers through this tool and obtain an MSIX package that you can

Insider programs offer a way to test new features of MSIX packaging. Microsoft is also working with multiple packaging tools available in the market like InstallShield, Cloudhouse, InstallAware and Advanced Installer to bring MSIX.

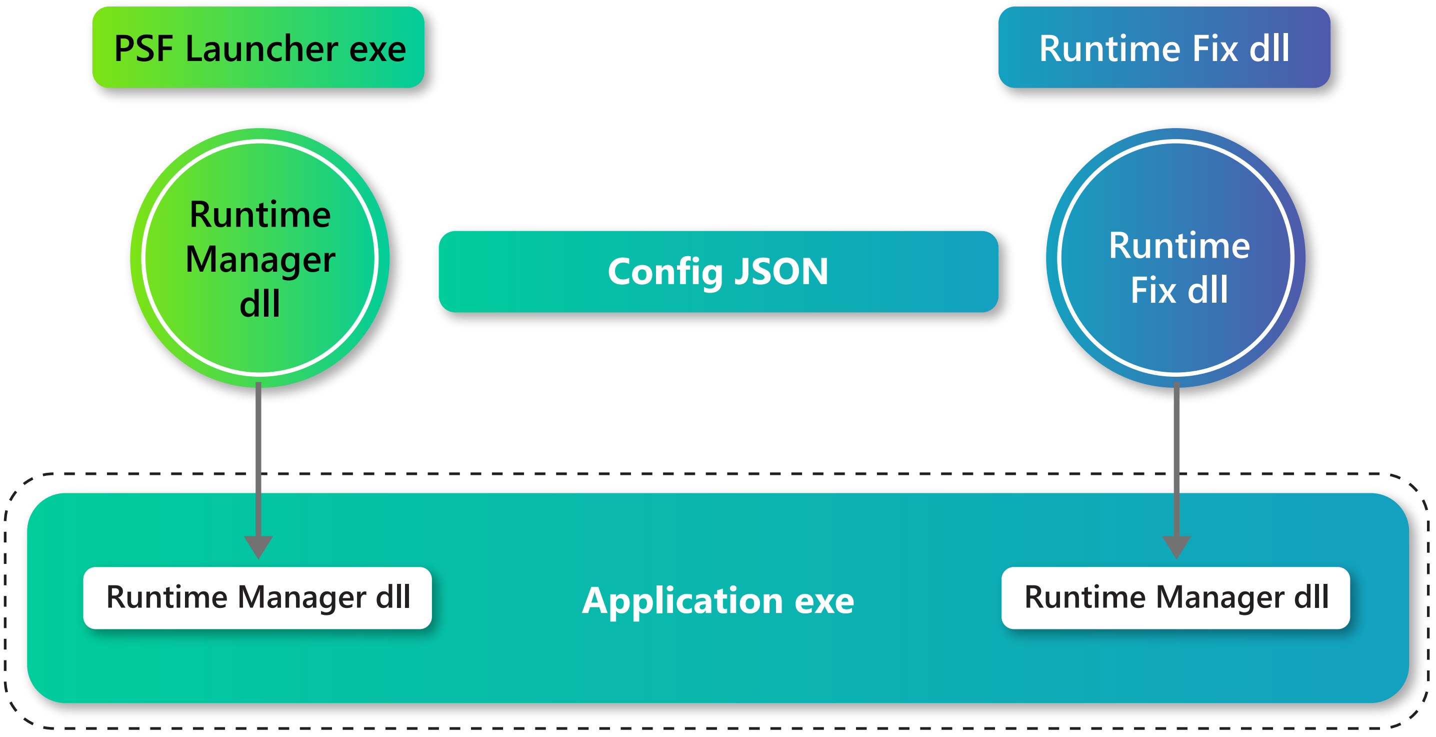
If you are a developer / re-packager, you will probably feel more comfortable doing the packaging in an environment, or you might already know, or using it every day for your development tasks. Visual Studio offers built-in MSIX packaging integration, making it easy to package an existing project built with WPF, Windows Forms, or Visual C++. Visual Studio has also gained the ability to package applications you do not own the source code. You will not have support for all the features, but you will leverage a better manifest editor and an integrated experience to generate an MSIX package.

Package SUPPORT FRAMEWORK

We still have differentiation between traditional Win32 and packaged one. There is a certain standard to follow to create a good application package. The MSIX package will not create a shortcut in the start menu, so APIs to retrieve the current working directory will return **C:\Windows\System32 or C:\Windows\SysWOW64** instead of the folder where the application has been deployed. This behaviour might be challenging for the application that tries to load files deployed with the executable in the same folder.

MSIX package might not be having permission to write the INSTALLDIR folder location, which is deployed inside the **C:\Program Files\WindowsApps folder**, which is a system protected. The best approach to solving this kind of problem is changing the code and fixing the wrong behaviour. However, this is not always possible. As an IT Professional, you might need to deploy applications from third-party developers, so you do need to have access to the source code; maybe it is an ancient application that is still widely used but no longer maintained.

The Package Support Framework is an open-source project published by Microsoft on GitHub. It will allow IT professionals to change the logic of the application without modifying the code.



The above diagram explains the architecture of the Package Support Framework.

A packaged application that uses the Package Support Framework is composed of four components:

- **Launcher:** it is executable
- **Runtime Manager:** which is a DLL
- JSON Configuration File
- One or more runtime fixes, which are DLLs that contain the logic to change the behaviour of the application

When the framework is applied to the packaged application, the launcher replaces the application's executable as the main entry point. The launcher will insist on executing the main application with the injected runtime manager. The JSON file controls the whole configuration.
The runtime fixes can change the workflow of Windows APIs.

Distribute

YOUR MSIX PACKAGE

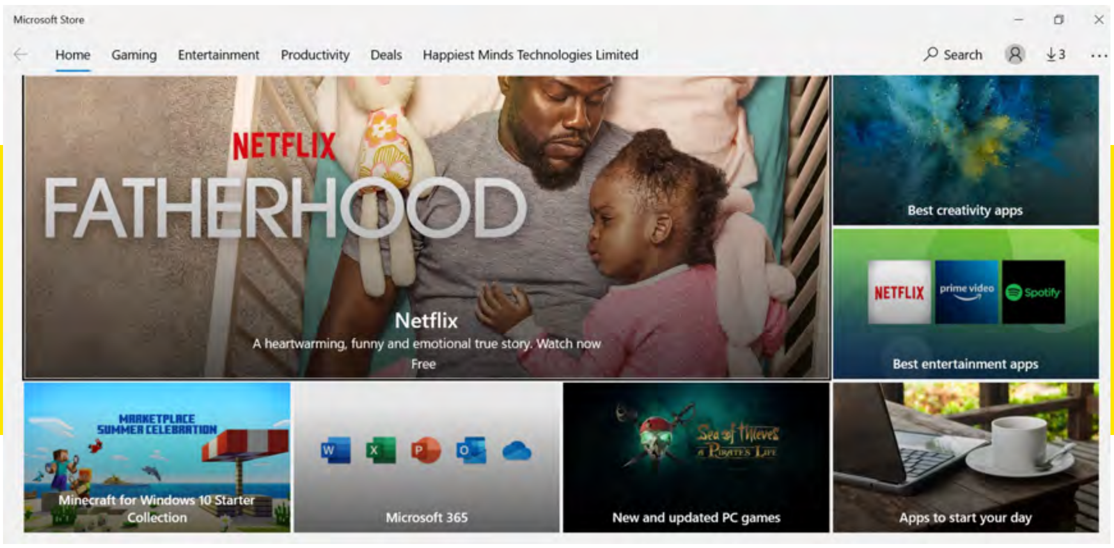
MSIX is more flexible in distributing the application. It employs the ability to deploy with a wide range of options like SCCM, Intune, a file share.

Example two different distribution approaches
- Microsoft Store and the web
- Windows 10 feature called App Installer

MICROSOFT STORE AND WEB DISTRIBUTION

For target consumer-based distribution Microsoft Store is a great platform. The Microsoft Store is built into Windows 10, and it provides users easy access to a comprehensive catalogue of applications.

Microsoft Store also opened the doors to classic desktop applications packaged as AppX or MSIX. Spotify, Slack, and iTunes are just some examples of Win32 applications easily downloaded from the store.



APP INSTALLER DISTRIBUTION

If an organization has not adopted the store for business for application deployment, the App installer feature in Windows 10 comes into play. This App Installer enables the store feature by deploying them on a website or file share.

Earlier, PowerShell was the only option for deploying a package without using the store, but now with the App Installer, you can double-click an MSIX package and proceed with the deployment.
App Installer has a strategy of MS-App Installer protocol, which can be used for website or file share to trigger the MSIX deployment. The application will start downloading when the user clicks the button.

Conclusion

MSIX brings new opportunities to deploy your application with a modern outlook and leverage it on existing technologies. It will work on the existing deployment method SCCM or Intune and work with the new method Microsoft Store, making the product more flexible than other methods available.

In the end, as a developer, you can combine the power of MSIX and its deployment flexibility to make it more agile and to introduce a DevOps approach in your projects. We can go from the source code to the deployment automatically, without any action from our side, other than committing new code to the repository where your project is hosted.

MSIX Labs Microsoft recently published an open-source project on GitHub called MSIX Labs. It offers many exercises and a step-by-step manual that covers the MSIX Packaging Tool.

Author

BIO

Yashaswi KC is a Digital Workspace Architect, and he has over 11+ years of IT experience with End User Computing Technology. He has spent many years planning, designing, consulting for different EUCS Platforms.

He has worked with multiple digital workspace platforms like MMD, WVD (AVD), Autopilot, Intune, Application packaging, Nexthink. Yashaswi KC is currently working as an Associate Architect at Happiest minds Technologies. He is responsible for Preparing and launching EUC related Go-to-Market strategies and service offerings to customers. Assist Sales team on RFPs (new and proactive deals) for EUC technologies and services.