



MSIX Legacy Application Launcher 1.0.0.2

The MSIX Legacy App Launcher is developed for software developers and software package engineers responsible for deploying their (legacy) applications using the new Microsoft MSIX packaging format. According to Microsoft, MSIX is a Windows app package format that provides a modern packaging experience to all Windows apps. The MSIX package format preserves the functionality of existing app packages and/or install files in addition to enabling new, modern packaging and deployment features to Win32, WPF, and Windows Forms apps.

MSIX enables enterprises to stay current and ensure their applications are always up to date. It allows IT Pros and developers to deliver a user centric solution while still reducing the cost of ownership of application by reducing the need to repackaging.

Some of the key benefits are:

- Reliability. MSIX provides a reliable install boasting a 99.96% success rate over millions of installs with a guaranteed uninstall.
- Network bandwidth optimization. MSIX decreases the impact to network bandwidth through downloading only the 64k block. This is done by leveraging the AppxBlockMap.xml file contained in the MSIX app package. MSIX is designed for modern systems and the cloud.
- Disk space optimizations. With MSIX there is no duplication of files across apps and Windows manages the shared files across apps. The apps are still independent of each other so updates will not impact other apps that share the file. A clean uninstall is guaranteed even if the platform manages shared files across apps.

Although the MSIX packaging format is still a work in progress, some (legacy) support will not be available using this new format. The MSIX Legacy App Launcher will solve the following issues when (re)packaging legacy applications for the Microsoft MSIX platform:

- Set the current working directory when the target executables are launched.
- Pass arguments to the target executables when launched.
- Start non-executable targets like weblinks or scripts.
- Set session environment variables for the launched target executables.
- Perform basic script actions before and after the targets are launched.
- Start external (non-containerized) executables inside the MSIX package.
- Use of relative (VFS or PVAD) paths for setting environments variables, performing script actions and launching the targets.

This tool can be used as a replacement or supplement for the Microsoft package support framework (PSF) to solve some of the common issues when (re)packaging legacy applications without the need for editing or rebuilding the MSIX package after its first creation.





Contents

MSIX Legacy Application Launcher 1.0.0.2.....	1
How to use.....	3
Deployment strategy.....	4
INI File usage	5
Explanation MAIN Section.....	5
Explanation APP Section.....	6
Explanation SetEnv Items.....	9
Explanation GetEnv Items	9
Explanation SCRIPT items.....	10
Environment Variables Explanation	11
Multiple Entry Points Explanation.....	14
Troubleshooting	16
Using PVAD or VFS.....	17
Developer information	18









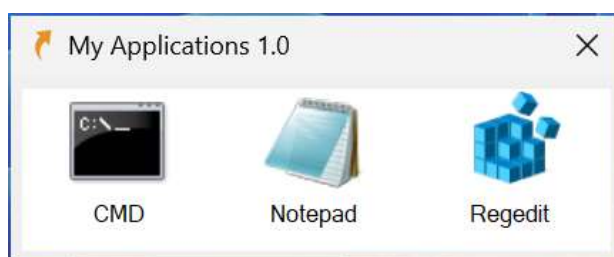
How to use

The MSIX Legacy App Launcher is a standalone single executable available in a 32-bit and 64-bit release. Use the 32-bit release for creating a 32-bit MSIX package that can be deployed on a 32-bit or 64-bit Windows 10/11 operating system and the 64-bit release for creating a 64-bit package that can be deployed on a 64-bit Windows 10/11 operating system. The executable is named *MSIXLegacyAppLauncher32.exe* and *MSIXLegacyAppLauncher64.exe* by default and can be renamed to whatever name you like. For example *MyMainApp1.0.exe*.

The renamed MSIX Legacy App Launcher executable needs its own corresponding INI file with the same name as the chosen executable name and must be in the same folder. For example, *MyMainApp1.0.exe* with *MyMainApp1.0.ini*.

Name	Date modified	Type	Size
 MSIXLegacyAppLauncher32	02/02/2023 11:26	Application	416 KB
 MSIXLegacyAppLauncher32	25/01/2023 16:36	Configuration settings	1 KB
 MSIXLegacyAppLauncher64	02/02/2023 11:26	Application	513 KB
 MSIXLegacyAppLauncher64	25/01/2023 16:36	Configuration settings	1 KB

Use the renamed MSIX Legacy App Launcher executable and place it with the corresponding INI file in the MSIX package (root) folder. Collect all shortcut information from the legacy application in the start menu and remove them. Create one or more shortcuts in the start menu to the renamed MSIX Legacy App Launcher executable(s) during capturing with the Microsoft MSIX Packaging Tool.



An example of the MSIX Legacy App Launcher main GUI that can be created by modifying the corresponding INI file. When using a single target (executable) the main GUI will not be shown and the single target (executable) will be launched directly.

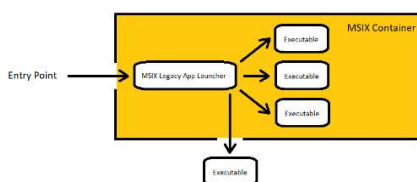


Deployment strategy

The (renamed) MSIX Legacy App Launcher executable can be used in various ways when (re)packaging legacy applications for the Microsoft MSIX platform. Depending on the (legacy) application being used, you can combine one or more of the following deployment methods.



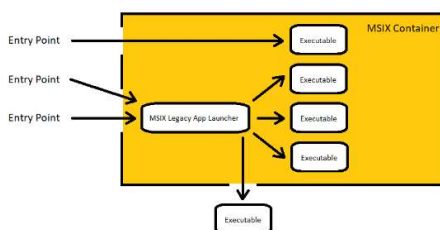
The single entry point executing a single target method will skip the main GUI and will directly start the target (executable).



The single entry point executing multiple targets method will display the main GUI and will start the chosen target (executable) from the main GUI.



*The multiple entry points executing a single target method will skip the main GUI and will directly start the target (executable). This will require editing the **AppxManifest.xml** file where all the Application “Id” name elements for each entry point must match with the corresponding name of the **App** key parameters (App1=, App2=, etc.) defined under the MAIN section of the INI-file. If no match is found for a specified entry point the main GUI will be shown for starting that entry point. (See Multiple Entry Points)*



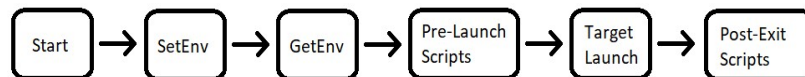
Combine the above methods to suit the needs of the (legacy) application to operate as required.



INI File usage

The corresponding INI File exist starting with the **MAIN** section that defines the number of shortcuts shown in the main graphical user interface (GUI) and one or more sub sections starting with **APP1**, **APP2**, **APP3**, etc. Each **APP** section contains the shortcut information with additional environment variables, pre-launch and post-exit script actions. If only one shortcut section is defined (APP1) the main graphical user interface (GUI) is skipped and the App1 section will be started immediately.

The execution order for each shortcut by the MSIX Legacy App Launcher process is always by executing the environment items **SetEnv** and **GetEnv** first, followed by the **Pre-Launch** script items, then the **target** (executable) item and lastly the **Post-Exit** script items.



Explanation MAIN Section

The **MAIN** section is required as a starting point to define all shortcuts and settings for creating the (Main) graphical user interface (GUI). Make sure this section and all the key parameters exist in the INI File.

The **MAIN** section uses the following key parameters:

Title

This is the title text shown in the main GUI title bar. For example: My Application 1.0

Icon

This is the title icon shown in the main GUI title bar. For example: .\Mylcon.ico
If omitted, the default icon will be shown.

Height

The main GUI height set in the number of pixels. If omitted the default is 200 pixels.

Width

The main GUI width set in the number of pixels. If omitted the default is 330 pixels.

ButtonOK

The "GetEnv GUI" OK button text to show. If omitted the default is set to the text OK. You can change this text for starting the target (executable) to the corresponding language text you like. (See Explanation GetEnv Items)

IconSize

The size of the icons shown in the main GUI List-View set in the number of pixels. If omitted the default will be 48 showing all icons in a 48x48 pixel resolution.

NoLog

If omitted, a logfile with the same name as the (renamed) MSIX Legacy App Launcher executable will be created in the personal temp folder each time when the MSIX Legacy App Launcher is executed and a shortcut icon is launched. If the *NoLog* parameter is set to 1 no logfile will be created. Only set this parameter to 1 if you don't want to perform troubleshooting anymore. (See Troubleshooting)

App1

The name of the first shortcut to be used with the first target executable.

App2

The name of the second shortcut to be used with the second target executable. You can continue adding more App options like App3, App4, etc. for adding more shortcuts in the main GUI.





Example:

```
[MAIN]
Title=My Applications 1.0
Icon=.\Mylcon.ico
Height=100
Width=300
ButtonOK=Start
IconSize=64
NoLog=
App1=CMD
App2=Notepad
App3=Regedit
```

Explanation APP Section

For each defined app key parameter in the MAIN section create a separate APP section. For example **[APP1]**, **[APP2]**, **[APP3]**, etc. Each created APP section contains all the information for setting environment variables, performing script actions and executing the target (executable).

Each **APP** section uses the following key parameters:

Icon

Specify an icon file (*.ico) or executable path with the starting icon number to show in the MSIX Legacy App Launcher main GUI ListView. For example: %ComSpec%,1

Target

This is the target executable to the (external) (legacy) program. For example: MyApplication.exe or %ComSpec%

WorkingDir

This is the working directory for the launched target item. For example: %A_WorkingDir%\Bin using PVAD or %A_ProgramFiles%\MyApp\Bin using VFS. (See Using PVAD or VFS)

Options

If omitted, the target executable launches normally. You can also use *Max*, *Min* or *Hide*. Max launches the target executable maximized, Min launches the target executable minimized and Hide launches the target executable hidden.

Param1

The first parameter to be used with the target executable. Like the **WorkingDir** parameter this can also contain relative directory paths. For example: %A_WorkingDir%\data\test.cfg

If you want to pass through arguments that are added during startup using the *MSIX Filetype Association* use **%1%** for the first argument and **%2%** for the second and so on. Quotes will be added automatically if spaces are detected in the parameter value when executing the target (executable).

Param2

The second parameter to be used with the target executable. You can continue adding more *param* options like Param3, Param4, etc. All *Param* property values will be joint together with a space in between when executing the Target executable.



**Wait**

If omitted, immediately after the target executable is launched the MSIX Legacy App Launcher process will log a failed or successful launch of the target executable. Post-Exit script actions will not be performed. If the *Wait* parameter is set to 1 the target executable is launched and the MSIX Legacy App Launcher process will wait for the target process until the target process is stopped where the target executable return code is logged and Post-Exit script actions can be executed.

SetEnv1

The first environment variable to set before performing pre-launch script actions and starting the target executable. See Explanation SetEnv Items for more information.

SetEnv2

The second environment variable to set before performing pre-launch script actions and starting the target executable. See Explanation SetEnv Items for more information. You can continue adding more *SetEnv* options like SetEnv3, SetEnv4, etc.

GetEnv1

The first environment variable to get from the user before performing pre-launch script actions and starting the target executable. See Explanation GetEnv Items for more information.

GetEnv2

The second environment variable to get from the user before performing pre-launch script actions and starting the target executable. See Explanation GetEnv Items for more information. You can continue adding more *GetEnv* options like GetEnv3, GetEnv4, etc.

PreLaunch1

The first pre-launch script action to perform before executing the main target. See Explanation SCRIPT items for more information.

PreLaunch2

The second pre-launch script action to perform before executing the main target. See Explanation SCRIPT items for more information. You can continue adding more *PreLaunch* options like PreLaunch3, PreLaunch4, etc.

PostExit1

The first post-exit script action to perform when the main target is stopped executing. See Explanation SCRIPT items for more information.

PostExit2

The second post-exit script action to perform when the main target is stopped executing. See Explanation SCRIPT items for more information. You can continue adding more *PostExit* options like PostExit3, PostExit4, etc.

RunInPackage

If omitted or zero (0), the (external) target executable will be launched directly. If the target executable is present in the MSIX package keep this value omitted or 0. Set the *RunInPackage* parameter only for launching external executables inside the MSIX context of the MSIX package.

If the *RunInPackage* parameter is set to 1 or 2 an external (non-containerized) target executable can be launched inside the MSIX context of the MSIX package using the Powershell *Invoke-CommandInDesktopPackage* cmdlet. If the parameter is set to 1 the Powershell process will run non-elevated and if the parameter is set to 2 the Powershell process will run elevated showing the User Access Control (UAC) window before launching. Use the run elevated option to start a processes that require elevation like Regedit.exe. Set the **Options** parameter to *Hide* if you don't want to see the Powershell window briefly come up when launching the target executable.





In order for the *RunInPackage* option to work correctly the MSIX Legacy App Launcher executable must be launched from inside the MSIX package. Keep in mind that environment variables set before launching the target executable will not be passed-on by the Powershell process for the target executable. Also, **Post-Exit** script action will be performed immediately after the Powershell.exe process is closed. This is before the target executable! External target executables that are already inside a containerized package (like Notepad.exe) cannot access the MSIX context of the MSIX package.

For more information about the Powershell Invoke-CommandInDesktopPackage module see:

<https://learn.microsoft.com/en-us/powershell/module/appx/invoke-commandindesktoppackage?view=windowsserver2022-psExample>

Example:

```
[APP1]
Icon=%ComSpec%,1
Target=%ComSpec%
WorkingDir=%A_WorkingDir%
Options=
Param1=
Wait=1
SetEnv1=MyWorkingDir %A_WorkingDir%
GetEnv1=UserName %A_UserName%
GetEnv2=Password 1
PreLaunch1=FileCopy, %A_WorkingDir%\test.txt, %A_AppData%, 1
PostExit1=FileDelete, %A_AppData%\test.txt
RunInPackage=0
```

```
[APP2]
Icon=Notepad.exe,1
Target=Notepad.exe
WorkingDir=
Options=
Param1=
Wait=0
RunInPackage=0
```

```
[App3]
Icon=Regedit.exe,1
Target=Regedit.exe
WorkingDir=
Options=Hide
Param1=
Wait=0
RunInPackage=2
```





Explanation SetEnv Items

Before executing script items and the target executable it is possible to set a number of environments variables that also can contain relative paths like the %A_WorkingDir% (for PVAD) or %A_ProgramFiles% (for VFS) variable that can be used for executing the basic script actions and the target process. No permanent environment variables will be set for the current user or system.

Start each line under the APP section with the key parameter *SetEnv* immediately followed by a line number starting from 1. (SetEnv1=, SetEnv2=, SetEnv3=, etc.)

The variable following the key parameter exist with the environment variable name and a TAB followed by the environment variable value. If the environment variable should not be translated using existing environment variables use another TAB after the environment value and add the number 1 so the environment variable value is set as a fixed string.

Example:

```
SetEnv1=PS1Folder      %A_WorkingDir%\Scripts\MyPSscripts
SetEnv2=MyEnv Test
SetEnv3=PATH %A_WorkingDir%\Folder1\Folder2;%PATH%
SetEnv4=AppVarName %APPVAR1% 1
```

Explanation GetEnv Items

Before executing script items and the target executable it is also possible to get a number of environments variables directly from the user when launching the target executable by showing the Get Environment GUI (GetEnv GUI) where all the collected *GetEnv* environment variables names are shown. These environment variables values can then be entered or changed by the user before executing the script items and the target executable.

These environment variables are set for executing the basic script actions and the target process. No permanent environment variables will be set for the current user or system.

Start each line under the APP section with the key parameter *GetEnv* immediately followed by a line number starting from 1. (GetEnv1=, GetEnv2=, GetEnv3=, etc.)

The variable following the key parameter exist with the environment variable name and a TAB followed by a predefined environment variable value or text if needed or you can keep this value empty. If the environment variable should not be visible like entering a password use another TAB after the predefined (empty) environment variable value or text and add the number 1 so the entered environment variable value is hidden.





Example:

```
GetEnv1=UserName      %A_UserName%
GetEnv2=Password      1
```

Keep in mind that all (password) environment variables names and values are being logged when the NoLog option is omitted or 0.

Explanation SCRIPT items

Before and after launching the target executable in the APP section it is possible to execute the following script items: FileCopy, FolderCopy, FolderCreate, FileDelete, FolderDelete, RegWrite, RegDelete, and Run

Start each line under the APP section with either the key parameter *PreLaunch* or *PostExit* immediately followed by a line number starting from 1. (PreLaunch1=, PreLaunch2=, PostExit1=, PostExit2=, etc.)

PreLaunch parameters will be executed before the target executable is launched and *PostExit* parameters will be executed when the launched target executable is stopped.

Make sure the Wait value is set to 1 in the APP section if you want to perform Post-Exit script actions.

The variable following the key parameter *PreLaunch* or *PostExit* exists with the script item name and a comma with a maximum of five item values also separated by a comma. The number of item values and what item values that are used are depending on the script item used as described below.

FileCopy, SourcePattern, DestPattern, 1 or 0 (If this parameter is 1, the command overwrites existing files)

FolderCopy, SourceFolder, Destination, 1 or 0 (If this parameter is 1, the command overwrites existing files and folders)

FolderCreate, DirName

FileDelete, FileName(s)

FolderDelete, DirName (Remove all files and subdirectories)

RegWrite, (REG_SZ, REG_EXPAND_SZ, REG_MULTI_SZ, REG_DWORD, or REG_BINARY), (HKEY_LOCAL_MACHINE, or HKEY_CURRENT_USER), ValueName, Value

RegDelete, (HKEY_LOCAL_MACHINE or HKEY_CURRENT_USER), ValueName (optional)

Run, Target, WorkingDir, (Min, Max or Hide), 1 or 0 (If this parameter is 1, the target command will be started without waiting for it to end.)

Keep in mind that writing to the MSIX packaging folder or HKEY_LOCAL_MACHINE registry is not allowed by default.

Example:

```
PreLaunch1=RegWrite, REG_BINARY, HKEY_CURRENT_USER\Software\TEST_APP, TEST_NAME,
01A9FF77
PreLaunch2=FolderCopy, %A_WorkingDir%\Personal Files, %A_MyDocuments%, 1
PreLaunch3=FolderCreate, %A_AppData%\%MyEnv%
PreLaunch4=FileCopy, %A_WorkingDir%\Personal Files\*.txt, %A_AppData%\TestApp, 0
PostExit1=RegDelete, HKEY_CURRENT_USER\Software\TEST_APP
PostExit2=FolderDelete, %A_Temp%\TempFolder
PostExit3=Run, %A_ComSpec% %A_WorkingDir%\MyScript.cmd, %A_WorkingDir%, Hide, 0
```





Environment Variables Explanation

For setting user environment variables, executing script items and starting the target executable, existing environment variables can be used in some of the key parameters values like %PATH%, %Temp%, %USERPROFILE%, etc.

The following **MSIX variables** are available when the MSIX Legacy App Launcher executable is launched from inside the MSIX package:

%PackagePath%

The full path of the MSIX package folder where the MSIX Legacy App Launcher is started from.

Example: C:\Program Files\WindowsApps\MSIXLegacyAppLauncher64_1.0.0.0_x64__cqqwa1hc8w2dp

%ManifestFile%

The full path and filename of the MSIX Manifest xml file (AppxManifest.xml) located in the PackagePath.

Example: C:\Program Files\WindowsApps\MSIXLegacyAppLauncher64_1.0.0.0_x64__cqqwa1hc8w2dp\AppxManifest.xml

%PackageName%

The name of the MSIX package where the MSIX Legacy App Launcher is started from.

Example: MSIXLegacyAppLauncher64

%PackageFullName%

The full name of the MSIX package where the MSIX Legacy App Launcher is started from.

Example: MSIXLegacyAppLauncher64_1.0.0.0_x64__cqqwa1hc8w2dp

%PublisherId%

The PublisherId of the MSIX package where the MSIX Legacy App Launcher is started from.

Example: cqqwa1hc8w2dp

%PackageFamilyName%

The family name of the MSIX package where the MSIX Legacy App Launcher is started from.

Example: MSIXLegacyAppLauncher64_cqqwa1hc8w2dp

%AppUserModelId%

The AppUserModelId of the MSIX Legacy App Launcher executable started from the MSIX package.

The value will depend on what entry point (shortcut) is used at startup.

Example: MSIXLegacyAppLauncher64_cqqwa1hc8w2dp!Regedit

%AppID% (See Multiple Entry Points Explanation)

The Application ID of the MSIX Legacy App Launcher executable started from the MSIX package. The value will depend on what entry point (shortcut) is used at startup.

Example: Regedit





The following build-in variable for the **MSIX Legacy App Launcher** can be used:

%FileName%

The (renamed) MSIX Legacy App Launcher executable and INI File are defined as followed:

%FileName%.exe and %FileName%.ini

The following build-in variables starting with "A_" can also be used:

%A_WorkingDir%

The full path of the directory where the MSIX Legacy App Launcher file is located.

%A_ScriptDir%

The full path of the directory where the MSIX Legacy App Launcher file is located.

%A_YYYY%

Current 4-digit year (e.g. 2023).

%A_MM%

Current 2-digit month (01-12).

%A_DD%

Current 2-digit day of the month (01-31).

%A_Hour%

Current 2-digit hour (00-23) in 24-hour time.

%A_Min%

Current 2-digit minute (00-59).

%A_Sec%

Current 2-digit second (00-59).

%A_Now%

The current local time in YYYYMMDDHH24MISS format.

%A_NowUTC%

The current Coordinated Universal Time (UTC) in YYYYMMDDHH24MISS format. UTC is essentially the same as Greenwich Mean Time (GMT).

%A_ComSpec%

Contains the same string as the environment's %ComSpec% variable. For example:

C:\Windows\system32\cmd.exe

%A_Temp%

The full path and name of the folder designated to hold temporary files.

%A_Language%

The system's default language. For example: 0413 = Dutch_Standard

%A_ComputerName%

The name of the computer as seen on the network.

%A_UserName%

The logon name of the user who launched this process.

%A_WinDir%

The Windows directory. For example: C:\Windows



**%A_ProgramFiles%**

The Program Files directory (e.g. C:\Program Files or C:\Program Files (x86)). This is usually the same as the ProgramFiles environment variable.

%A_AppData%

The full path and name of the folder containing the current user's application-specific data. For example: C:\Users\<UserName>\AppData\Roaming

%A_AppDataCommon%

The full path and name of the folder containing the all-users application-specific data. For example: C:\ProgramData

%A_Desktop%

The full path and name of the folder containing the current user's desktop files. For example: C:\Users\<UserName>\Desktop

%A_DesktopCommon%

The full path and name of the folder containing the all-users desktop files. For example: C:\Users\Public\Desktop

%A_StartMenu%

The full path and name of the current user's Start Menu folder. For example: C:\Users\<UserName>\AppData\Roaming\Microsoft\Windows\Start Menu

%A_StartMenuCommon%

The full path and name of the all-users Start Menu folder. For example: C:\ProgramData\Microsoft\Windows\Start Menu

%A_Programs%

The full path and name of the Programs folder in the current user's Start Menu. For example: C:\Users\<UserName>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs

%A_ProgramsCommon%

The full path and name of the Programs folder in the all-users Start Menu. For example: C:\ProgramData\Microsoft\Windows\Start Menu\Programs

%A_MyDocuments%

The full path and name of the current user's "My Documents" folder.



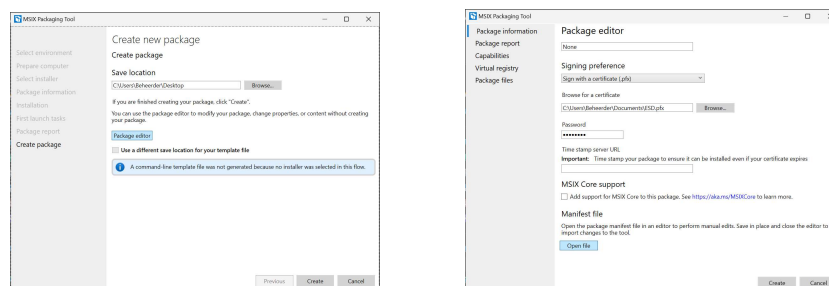


Multiple Entry Points Explanation

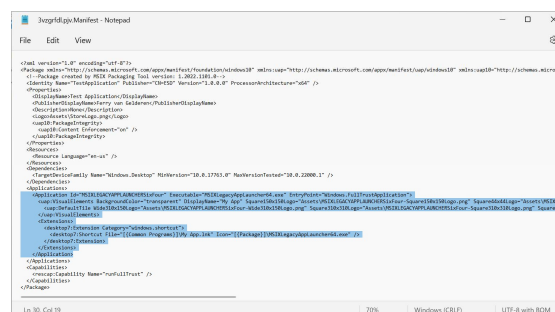
When multiple entry points are used for the MSIX package and the use of the main GUI is not desired when performing the MSIX Legacy App Launcher fix-ups for each target (executable) you can edit the **AppxManifest.xml** file so each entry point will launch the desired target (executable) using the single MSIX Legacy App Launcher executable with its corresponding INI-file that is present in the MSIX package.



For this to work you can edit the **AppxManifest.xml** file during the MSIX Packaging capture process using the Microsoft MSIX Packaging Tool. If the MSIX capture process is finished select the **Package editor** button to open the MSIX package.



On the “Package information” tab scroll down and click the **Open file** button.



Between the **<Applications>** and **</Applications>** elements all entry points are defined for each **Application** element. You can copy and paste multiple entry points by selecting an existing entry point starting with the line **<Application Id=** and all lines below including the **</Application>** line at the end.

Make sure that each entry point has its own unique **<Application Id=** name and **DisplayName=** visual element name. Make sure the entry points are starting the MSIX Legacy App Launcher executable by checking the **Executable=** value. Change each **<Application Id=** name to the corresponding **App** entry property name that is defined in the **MAIN** section of the MSIX Legacy App Launcher INI file so the MSIX Legacy App Launcher executable will know which target (executable) to start when it is launched using the installed entry point. If no match is found for a specified entry point the main GUI will be shown.



Example AppxManifest.xml file (Applications element):

```
<Applications>
  <Application Id="GUI" Executable="MSIXLegacyAppLauncher64.exe" EntryPoint="Windows.FullTrustApplication">
    <uap:VisualElements BackgroundColor="transparent" DisplayName="Start GUI"
Square150x150Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square150x150Logo.png"
Square44x44Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square44x44Logo.png" Description="Start CMD">
      <uap:DefaultTile Wide310x150Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Wide310x150Logo.png"
Square310x310Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square310x310Logo.png"
Square71x71Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square71x71Logo.png" />
    </uap:VisualElements>
    <Extensions></Extensions>
  </Application>
  <Application Id="CMD" Executable="MSIXLegacyAppLauncher64.exe" EntryPoint="Windows.FullTrustApplication">
    <uap:VisualElements BackgroundColor="transparent" DisplayName="Start CMD"
Square150x150Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square150x150Logo.png"
Square44x44Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square44x44Logo.png" Description="Start CMD">
      <uap:DefaultTile Wide310x150Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Wide310x150Logo.png"
Square310x310Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square310x310Logo.png"
Square71x71Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square71x71Logo.png" />
    </uap:VisualElements>
    <Extensions></Extensions>
  </Application>
  <Application Id="Notepad" Executable="MSIXLegacyAppLauncher64.exe" EntryPoint="Windows.FullTrustApplication">
    <uap:VisualElements BackgroundColor="transparent" DisplayName="Start Notepad"
Square150x150Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square150x150Logo.png"
Square44x44Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square44x44Logo.png" Description="Start CMD">
      <uap:DefaultTile Wide310x150Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Wide310x150Logo.png"
Square310x310Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square310x310Logo.png"
Square71x71Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square71x71Logo.png" />
    </uap:VisualElements>
    <Extensions></Extensions>
  </Application>
  <Application Id="Regedit" Executable="MSIXLegacyAppLauncher64.exe" EntryPoint="Windows.FullTrustApplication">
    <uap:VisualElements BackgroundColor="transparent" DisplayName="Start Regedit"
Square150x150Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square150x150Logo.png"
Square44x44Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square44x44Logo.png" Description="Start CMD">
      <uap:DefaultTile Wide310x150Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Wide310x150Logo.png"
Square310x310Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square310x310Logo.png"
Square71x71Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square71x71Logo.png" />
    </uap:VisualElements>
    <Extensions></Extensions>
  </Application>
</Applications>
```

Example of the corresponding INI file (MAIN section):

```
[MAIN]
Title=My Applications 1.0
Height=75
Width=275
ButtonOK=Start
IconSize=48
NoLog=
App1=CMD
App2=Notepad
App3=Regedit
```

The currently used Application Id value in the *AppxManifest.xml* file will be stored in the variable %AppID% when the MSIX Legacy App Launcher is started using the entry point. You can also check the log-file when starting the MSIX Legacy App Launcher and search for the line starting with "MSIX Appid". (See Troubleshooting)





Troubleshooting

When executing the MSIX Legacy App Launcher file and the **NoLog** property is omitted or 0 a log file will be created in the personal temp folder with the name of the (renamed) MSIX Legacy App Launcher executable. For example: *MyApp.log* when using *MyApp.exe* with *MyApp.ini*.

If the MSIX Legacy App Launcher is started from the MSIX package environment the logfile will also contain the Application Id (AppID) that will be added to the logfile name. For example: *MyApp_Regedit.log*

The log file is being overwritten each time the MSIX Legacy App Launcher is executed and it will provide all the information during runtime. Each logfile contains a line buildup starting at the top with a timestamp followed by an *Info*, *Warning* or *Error* statement and details on the performed action that involves getting the MSIX environment variables at startup, setting environment variables, executing the script items, and starting the target executable.

Example logfile:

```
20230208134109 Info MSIX Legacy Application Launcher 1.0.0.2 started for username: "Beheerder" on computer
name: "WIN-11-NL"
20230208134109 Info Operating System version: 10.0.22621 (64-bit=1)
20230208134109 Info Main working directory: C:\Program
Files\WindowsApps\MSIXLegacyAppLauncher64_1.0.0.2_x64__cqqlwa1hc8w2dp
20230208134109 Info Main executable name: MSIXLegacyAppLauncher64.exe
20230208134109 Info Main INI file name: MSIXLegacyAppLauncher64.ini
20230208134109 Info MSIX PackagePath: C:\Program
Files\WindowsApps\MSIXLegacyAppLauncher64_1.0.0.2_x64__cqqlwa1hc8w2dp
20230208134109 Info MSIX ManifestFile: C:\Program
Files\WindowsApps\MSIXLegacyAppLauncher64_1.0.0.2_x64__cqqlwa1hc8w2dp\AppxManifest.xml
20230208134109 Info MSIX AppxManifest.xml file found.
20230208134109 Info MSIX PackageFullName: MSIXLegacyAppLauncher64_1.0.0.2_x64__cqqlwa1hc8w2dp
20230208134109 Info MSIX PackageName: MSIXLegacyAppLauncher64
20230208134109 Info MSIX PublisherId: cqqlwa1hc8w2dp
20230208134109 Info MSIX PackageFamilyName: MSIXLegacyAppLauncher64_cqqlwa1hc8w2dp
20230208134109 Info MSIX AppUserModelId: MSIXLegacyAppLauncher64_cqqlwa1hc8w2dp!Regedit
20230208134109 Info MSIX AppId: Regedit
20230208134109 Info MSIX Application Id match found for: Regedit
20230208134109 Info Running App3: Regedit...
20230208134109 Info Create "REG_SZ" registry item "InstallDir" in "HKEY_CURRENT_USER\Software\MSIX Legacy App
Launcher" with value "C:\Program Files\WindowsApps\MSIXLegacyAppLauncher64_1.0.0.2_x64__cqqlwa1hc8w2dp"...
20230208134109 Info "HKEY_CURRENT_USER\Software\MSIX Legacy App Launcher\InstallDir" exist.
20230208134109 Info Create "REG_DWORD" registry item "Installed" in "HKEY_CURRENT_USER\Software\MSIX
Legacy App Launcher" with value "1"...
20230208134109 Info "HKEY_CURRENT_USER\Software\MSIX Legacy App Launcher\Installed" exist.
20230208134109 Info Create "REG_SZ" registry item "TimeStamp" in "HKEY_CURRENT_USER\Software\MSIX Legacy
App Launcher" with value "20230208134109"...
20230208134109 Info "HKEY_CURRENT_USER\Software\MSIX Legacy App Launcher\TimeStamp" exist.
20230208134109 Info Target=Regedit.exe
20230208134109 Info WorkingDir=
20230208134109 Info Options=Hide
20230208134109 Info Wait=0
20230208134109 Info RunInPackage=2
20230208134109 Info Starting Target...
20230208134109 Info Run (Elevated): powershell.exe Invoke-CommandInDesktopPackage -AppId 'Regedit' -
PackageFamilyName 'MSIXLegacyAppLauncher64_cqqlwa1hc8w2dp' -Command 'Regedit.exe' -Args '' -PreventBreakaway
20230208134110 Info Target successfully started with process ID: 6400
```





Using PVAD or VFS

When capturing an installation using the **Microsoft MSIX Packaging Tool** you can decide if you want to use the Virtual File System (**VFS**) only or the Primary Virtual Application Directory (**PVAD**) for your installation location to use. The VFS is still available when using the PVAD option but the folder structure set in the installation location during capturing will be made available in the MSIX package root folder where it isn't part of the VFS. The VFS option is used at default when the "Installation location" field is kept empty creating the package.

MSIX Packaging Tool

Create new package

Package information

Package name *

Package display name *

Publisher name *

Publisher display name *

Version *

Package Description

Installation location

Browse...

☐ Add support for MSIX Core to this package. See <https://aka.ms/MSIXCore> to learn more.

Once you continue to the next page, installation will begin and you will not be able to return.

Previous Next Cancel

For legacy applications it is recommended to use the VFS location to insure the original installation (VFS) path exist for all the application processes running inside the MSIX container. However, there are cases where using the PVAD location is needed to ensure a correctly working (legacy) application.

A best practice for (re)packaging a legacy application using the MSIX Legacy App Launcher is to place the MSIX Legacy App Launcher executable(s) with their corresponding INI file(s) in the "C:\MyApp" folder and set the "installation location" during the MSIX capturing process to this folder to ensure the MSIX Legacy App Launcher executable(s) with their corresponding INI file(s) become available in the MSIX Package root folder when finishing the capture process. Also, create the necessary shortcuts from the Start Menu pointing to the MSIX Legacy App Launcher executable(s) in the C:\MyApp location. The legacy software itself can then be installed either in their normal VFS location like "C:\Program Files\MyLegacyApp" or in the "C:\MyApp" folder for using the PVAD location.





Developer information

This software is provided "as is". Use of the software is free and at your own risk.

This software is created in AutoHotKey and the source code is available for review and/or adjustments.

Learn more about AutoHotKey:

<https://www.autohotkey.com>

Learn more about MSIX:

<https://learn.microsoft.com/en-us/windows/msix/overview>

Contact information:

Name: Ferry van Gelderen

Email: Ferry@Provolve.nl

Copyright © 2023 Provolve IT B.V.

