



MSIX Helper 1.0

The MSIX Helper 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 repackage.

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 Helper 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.
- Start external executables that are not part of the MSIX package environment.
- 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 environment.
- 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.





Contents

MSIX Helper 1.0	1
How to use.....	3
Implementing the MSIX Helper	3
INI File usage	6
Explanation Environment Variables Section	7
Explanation PreLaunch and PostExit SCRIPT Section	8
Run an external process in the context of the MSIX package	9
Troubleshooting	10
Using PVAD or VFS.....	12
Developer information	13









How to use

The MSIX Helper 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 and/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 *MSIXHelper32.exe* and *MSIXHelper64.exe* by default.

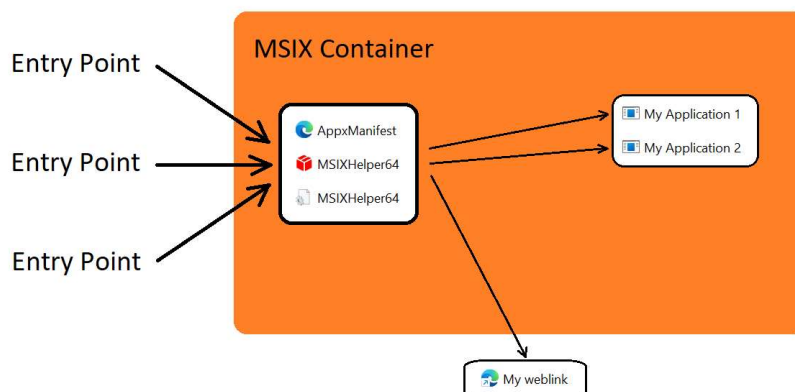
The MSIX Helper executable needs its own corresponding INI file with the same name and must be in the same folder. For example, *MSIXHelper64.exe* with *MSIXHelper64.ini*.

Name	Date modified	Type	Size
 MSIXHelper32.exe	14/06/2023 10:51	Application	455 KB
 MSIXHelper32.ini	06/06/2023 14:12	Configuration settings	2 KB
 MSIXHelper64.exe	14/06/2023 10:51	Application	559 KB
 MSIXHelper64.ini	06/06/2023 14:12	Configuration settings	2 KB

The MSIX Helper executable will only work correctly if it is started from the MSIX packaging environment using an entry point (AppX/MSIX shortcut or filetype association). When creating a MSIX Package copy the MSIX Helper executable and place it with the corresponding INI file in one of the available MSIX package (root or sub) folder.

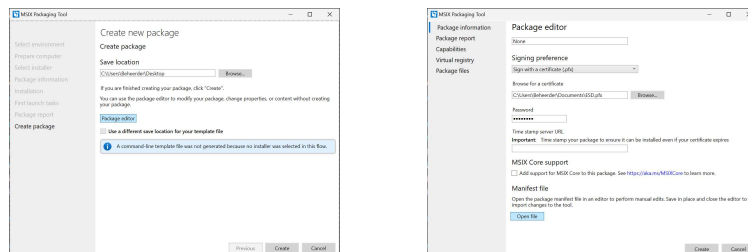
Implementing the MSIX Helper

The MSIX Helper executable will need to know what entry point (AppX/MSIX shortcut or filetype association) is used and what corresponding target (executable) for this entry point is used. This will require editing the AppxManifest.xml and/or INI-file where all the Application “Id” name elements for each entry point must be created as a separate section in the INI file. Edit the **AppxManifest.xml** and/or INI-file so each entry point will launch the desired target (executable) using the single MSIX Helper executable with its corresponding INI-file that is present in the MSIX package.

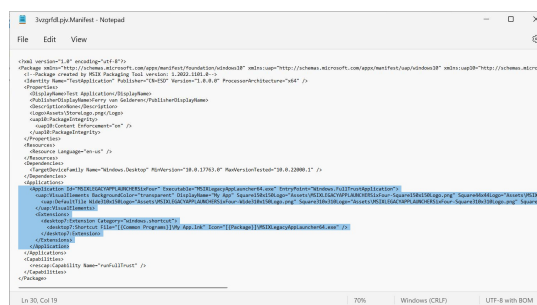




For this to work you can edit the **AppxManifest.xml** and/or **INI-file** during the MSIX Packaging capture process using the Microsoft MSIX Packaging Tool. If the MSIX capture process is finished select the **Package editor** button to open the MSIX package.



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 will have its own unique **<Application Id=** name and **DisplayName=** visual element name. Make sure the entry points are starting the MSIX Helper executable (MSIXHelper32.exe or MSIXHelper64.exe) by checking the **Executable=** value. Change each **<Application Id=** name to the corresponding name of the application section that is defined in the MSIX Helper INI file (or vice versa) so the MSIX Helper executable will know what target (executable) to start when it is launched using the specified entry point.

Keep in mind that for the AppxManifest.xml file the Application Id value can contain ONLY ALPHA-NUMERIC characters and you cannot use dash characters (-).



Example AppxManifest.xml file (Applications element):

```
<Applications>
  <Application Id="ApplicationOne" Executable="MSIXHelper64.exe" EntryPoint="Windows.FullTrustApplication">
    <uap:VisualElements BackgroundColor="transparent" DisplayName="Application 1" Square150x150Logo="Assets\App-Square150x150Logo.png"
      Square44x44Logo="Assets\App-Square44x44Logo.png" Description="Application 1">
      <uap:DefaultTile Wide310x150Logo="Assets\App-Wide310x150Logo.png" Square310x310Logo="Assets\App-Square310x310Logo.png"
        Square71x71Logo="Assets\App-Square71x71Logo.png" />
    </uap:VisualElements>
    <Extensions>
    </Extensions>
  </Application>
  <Application Id="ApplicationTwo" Executable="MSIXHelper64.exe" EntryPoint="Windows.FullTrustApplication">
    <uap:VisualElements BackgroundColor="transparent" DisplayName="Application 2" Square150x150Logo="Assets\App-Square150x150Logo.png"
      Square44x44Logo="Assets\App-Square44x44Logo.png" Description="Application 2">
      <uap:DefaultTile Wide310x150Logo="Assets\App-Wide310x150Logo.png" Square310x310Logo="Assets\App-Square310x310Logo.png"
        Square71x71Logo="Assets\App-Square71x71Logo.png" />
    </uap:VisualElements>
    <Extensions>
    </Extensions>
  </Application>
  <Application Id="ApplicationThree" Executable="MSIXHelper64.exe" EntryPoint="Windows.FullTrustApplication">
    <uap:VisualElements BackgroundColor="transparent" DisplayName="Application 3" Square150x150Logo="Assets\App-Square150x150Logo.png"
      Square44x44Logo="Assets\App-Square44x44Logo.png" Description="Application 3">
      <uap:DefaultTile Wide310x150Logo="Assets\App-Wide310x150Logo.png" Square310x310Logo="Assets\App-Square310x310Logo.png"
        Square71x71Logo="Assets\App-Square71x71Logo.png" />
    </uap:VisualElements>
    <Extensions>
    </Extensions>
  </Application>
</Applications>
```

Example of the corresponding MSIX Helper INI file:

```
[ApplicationOne]
Target=MyApplication1.exe
Options=
WorkingDir=%WorkingDir%
SetEnv=Environment
PreLaunch=PreLaunchScript
PostExit=PostExitScript

[ApplicationTwo]
Target=MyApplication2.exe
Param1=-h
WorkingDir=%WorkingDir%
SetEnv=Environment
PreLaunch=PreLaunchScript
PostExit=PostExitScript

[ApplicationThree]
Target=https://provolve.nl/

[Environment]
PATH=%WorkingDir%;PATH
AppBIN=%WorkingDir%\Bin

[PreLaunchScript]
FolderCreate=%LOCALAPPDATA%\MyApp
StringReplace=%WorkingDir%\conf.cfg, <AppData>, %APPDATA%, %LOCALAPPDATA%\MyApp\conf.cfg, 1
RegWrite=REG_SZ, HKEY_CURRENT_USER\Software\MyApp, WorkingDir, %WorkingDir%

[PostExitScript]
RegDelete=HKEY_CURRENT_USER\Software\MyApp
```





INI File usage

The corresponding INI File needs to have a separate section for each Application Id that is defined in the AppxManifest.xml. Each Application Id section will have the following key parameters:

Target

This is the target name (executable) to the (external) (legacy) program. For example: MyApplication.exe or %ComSpec% using target executables or use a weblink like <https://www.google.com> or <mailto:someone@somedomain.com> to open the default e-mail application with the recipient filled in. Add **RunAs* (and a space) before the target executable name to force elevation (*Run as administrator*).

WorkingDir

This is the working directory for the launched target item. For example: %WorkingDir%\Bin using PVAD or %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: %WorkingDir%\data\test.cfg

Quotes (") will be added automatically before and after the parameter value if spaces are detected in the parameter value.

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 joined together with a space in between when executing the **Target** executable. MSIX Filetype Association arguments will be added automatically after the joined Parameter property values.

SetEnv

The name of the environment variable section that will be executed before performing the *PreLaunch* script actions and starting the target (executable). See Explanation Environment Variables Section for more information.

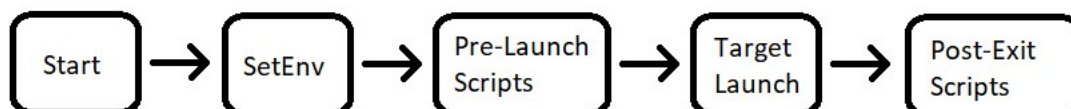
PreLaunch

The name of the *PreLaunch* script section that will be executed before executing the main target (executable). See Explanation PreLaunch and PostExit SCRIPT Section for more information.

PostExit

The name of the *PostExit* script section that will be executed after the main target (executable) is stopped executing. See Explanation PreLaunch and PostExit SCRIPT Section for more information.

The execution order for each Application Id target by the MSIX Helper process is always by executing the environment variables items **SetEnv** first, followed by the **PreLaunch** script items, then the **target** (executable) item and lastly the **PostExit** script items when these are used.





Example:

```
[ApplicationOne]
Target=MyApplication1.exe
Options=
Param1=
WorkingDir=%WorkingDir%
SetEnv=Environment
PreLaunch=PreLaunchScript
PostExit=PostExitScript
```

```
[ApplicationTwo]
Target=MyApplication2.exe
Options=
Param1=
WorkingDir=%WorkingDir%
SetEnv=Environment
PreLaunch=PreLaunchScript
PostExit=PostExitScript
```

```
[ApplicationThree]
Target=https://provolve.nl/
```

Explanation Environment Variables Section

When the MSIX Helper is started from the MSIX package environment the following environment variables will be available for the MSIX Helper process and all child (target) processes:

WorkingDir

This will contain the full path where the MSIX Helper is located and started from.

PackagePath

This will contain the full path of the MSIX package root folder where the MSIX Helper is started from.

Example: C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64__cqqlwa1hc8w2dp

PackageFamilyName

This will contain the family name of the MSIX package where the MSIX Helper is started from.

Example: MSIXHelper64_cqqlwa1hc8w2dp

AppId

This will contain the currently used Application Id of the MSIX Helper executable started from the MSIX package. The value will depend on what entry point (shortcut) is used at startup.

Before executing script items and the target executable it is possible to set a number of environment variables that also can contain relative paths like the %WorkingDir% (for PVAD) or %ProgramFiles% (for VFS). These environment variables are available for executing the *PreLaunch* and *PostExit* script actions and the target executable. Start each line under the environment section name with the environment variable name and an equal sign (=) followed by the environment variable value.

Example:

```
[Environment]
ENV=Production
PATH=%WorkingDir%;%PATH%
AppBIN=%WorkingDir%\Bin
```

All environment variables that are defined under the section will be set starting at the top until the end of the section is reached. You can create more environment sections for other targets that need other environment variables at startup. No permanent environment variables will be set for the current user or system.





Explanation PreLaunch and PostExit SCRIPT Section

Before and after launching the target (executable) it is possible to execute the following script items:

FileCopy, FolderCopy, FolderCreate, FileDelete, FolderDelete, RegWrite, RegDelete, Run, StringReplace, FileAppend and Download.

Start each line under the *PreLaunch* and/or *PostExit* section name with the script item name and an equal sign (=) followed by a maximum of five item values 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.

Script Item	Item Value 1	Item Value 2	Item Value 3	Item Value 4	Item Value 5
FileCopy	Source (wildcard) Pattern	Destination (wildcard) Pattern	1 (overwrites) 0		
FolderCopy	Source Folder	Destination path	1 (overwrites) 0		
FolderCreate	Folder name path				
FileDelete	File(s) (wildcard) Pattern				
FolderDelete	Folder name path (Deletes all files and subfolder)				
RegWrite	REG_SZ REG_EXPAND_SZ REG_MULTI_SZ REG_DWORD REG_BINARY	HKEY_LOCAL_MACHINE HKEY_CURRENT_USER	Value name	Value (If the value doesn't exist the value will be created)	
RegDelete	HKEY_LOCAL_MACHINE HKEY_CURRENT_USER	Value name (optional)			
Run	Target	WorkingDir	Min Max Hide (If omitted, the target executable launches normally)	0 (wait for target command to end) 1 (the target command will be started without waiting for it to end)	
StringReplace	Source Filename (If Source and Destination filename are equal, multiple String Replacements can be done for the same file and Overwrite will be ignored)	Name of the text string to find	Name of the text to replace with (All occurrences of the search text will be replaced)	Destination filename (If the file doesn't exist the file will be created first and the location needs to be writable)	1 (overwrites) 0 (Skip StringReplace when destination file already exists)
FileAppend	Text write to the end of a file	Full file name path (If the file doesn't exist the file will be created first and the location needs to be writable)	UTF-8 UTF-8-RAW UTF-16 UTF-16-RAW (If omitted the default encoding will be ANSI)		
Download	URL (https/ftp)	Full file name path (Any existing file will be overwritten)			

Variables (%) can be used instead of fixed texts for file (paths) and registry values. (See example below)

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

Script items set under the *PreLaunch* section name will be executed before the target executable is launched and script items set under the *PostExit* section name will be executed when the launched target executable is stopped.





Example:

```
[PreLaunchScript]
FileCopy=%WorkingDir%\Document.txt, %APPDATA%, 1
FolderCopy=%WorkingDir%\Resource, %USERPROFILE%\Documents\Resource, 1
FolderCreate=%APPDATA%\%COMPUTERNAME%
FileAppend=Another line, %APPDATA%\test.txt
FileAppend=%WorkingDir%, %APPDATA%\test.txt
StringReplace=%WorkingDir%\conf.cfg, <AppData>, %APPDATA%, %LOCALAPPDATA%\MyApp\conf.cfg, 0
Download=https://www.autohotkey.com/download/2.0/version.txt, %LOCALAPPDATA%\version.txt
RegWrite=REG_SZ, HKEY_CURRENT_USER\Software\MyApp, WorkingDir, %WorkingDir%
Run=Notepad.exe "%WorkingDir%\Document.txt", %WorkingDir%, 0

[PostExitScript]
FileDelete=%APPDATA%\Document.txt
FolderDelete=%USERPROFILE%\Documents\Resource
FolderDelete=%APPDATA%\%COMPUTERNAME%
FileDelete=%APPDATA%\test.txt
FileDelete=%LOCALAPPDATA%\version.txt
RegDelete=HKEY_CURRENT_USER\Software\MyApp, WorkingDir
RegDelete=HKEY_CURRENT_USER\Software\MyApp
```

Run an external process in the context of the MSIX package

Use the Powershell Cmdlet **Invoke-CommandInDesktopPackage** to start an external process in the context of the MSIX package environment. The created process will have the identity of the provided AppId and will have access to its virtualized file system and registry (if any). The new process will have a token that is similar to, but not identical to, a real AppId process. The primary use-case of this command is to invoke debugging or troubleshooting tools in the context of the packaged app to easily access its virtualized resources. For example, you can run the Registry Editor to see virtualized registry keys.

The following example will start the Regedit process elevated using the MSIX Helper INI-File Application Id section:

```
[REGEDIT]
Target=*RunAs powershell.exe
WorkingDir=
Options=Hide
Param1=Invoke-CommandInDesktopPackage
Param2=-AppId
Param3=%AppId%
Param4=-PackageFamilyName
Param5=%PackageFamilyName%
Param6=-Command
Param7=Regedit.exe
Param8=-PreventBreakaway
```

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, *PostExit* script action will be performed immediately after the Powershell.exe process is closed.

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

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





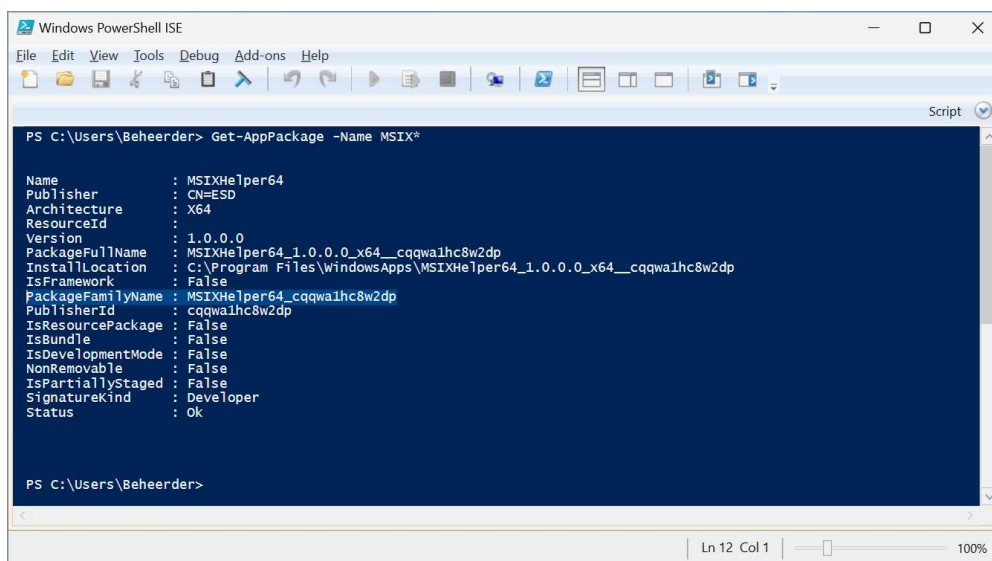
Troubleshooting

When executing the MSIX Helper executable at default no log file will be created.

If you want to use a log file for troubleshooting first make sure the MSIX Helper executable is started from the MSIX Package environment. You then need to manually create an empty file in the personal temp folder that has the *PackageFamilyName* property name and the .log extension.

For example: %TEMP%\MSIXHelper64_cqqlwa1hc8w2dp.log

You can use the Powershell cmdlet *Get-AppPackage* to find the *PackageFamilyName* value for your package name.



```
PS C:\Users\Beheerder> Get-AppPackage -Name MSIX*

Name                : MSIXHelper64
Publisher            : CN=ESD
Architecture        : X64
ResourceId           :
Version             : 1.0.0.0
PackageFullName      : MSIXHelper64_1.0.0.0_x64__cqqlwa1hc8w2dp
InstallLocation      : C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64__cqqlwa1hc8w2dp
IsFramework          : False
PackageFamilyName    : MSIXHelper64_cqqlwa1hc8w2dp
PublisherId          : cqqlwa1hc8w2dp
IsResourcePackage    : False
IsBundle             : False
IsDevelopmentMode    : False
NonRemovable         : False
IsPartiallyStaged    : False
SignatureKind        : Developer
Status               : Ok

PS C:\Users\Beheerder>
```

The log file 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* level statement and details on the performed action that involves setting environment variables, executing the script items, and starting the target executable. Logging will continue as long as the log file exist and will stop when the log file is deleted.





Example logfile:

Timestamp	Level	Details
20230328193124	[INFO]	MSIX Helper 1.0 started for Application Id: notepad
20230328193124	[INFO]	Computer name: HPOMEN16
20230328193124	[INFO]	Operating System version: 10.0.22621 (64-bit=1)
20230328193124	[INFO]	Operating System language code: 0413
20230328193124	[INFO]	User name: FerryvanGelderden
20230328193124	[INFO]	Working directory: C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64_cqqa1hc8w2dp
20230328193124	[INFO]	MSIX PackagePath: C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64_cqqa1hc8w2dp
20230328193124	[INFO]	MSIX PackageFamilyName: MSIXHelper64_cqqa1hc8w2dp
20230328193125	[INFO]	MSIX AppId: notepad
20230328193125	[INFO]	[Setting Environment Variables for section: Environment]
20230328193125	[INFO]	Environment variable successfully set for 'MyName' with value: Ferry van Gelderen
20230328193125	[INFO]	Environment variable successfully set for 'MyWorkingDir' with value: C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64_cqqa1hc8w2dp
20230328193125	[INFO]	Environment variable successfully set for 'MyFixedVar' with value: %Ferry%
20230328193125	[INFO]	Environment variable successfully set for 'MyComputer' with value: HPOMEN16
20230328193125	[INFO]	[Performing PreLaunch script section: PreLaunchScript]
20230328193125	[INFO]	FileCopy from source C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64_cqqa1hc8w2dp\Document.txt' to destination 'C:\Users\FerryvanGelderden\AppData\Roaming' (OverWrite=1)
20230328193125	[INFO]	FileCopy successfully executed.
20230328193125	[INFO]	FolderCopy from source C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64_cqqa1hc8w2dp\Resource' to destination 'C:\Users\FerryvanGelderden\Documents\Resource' (OverWrite=1)
20230328193125	[INFO]	FileCopy successfully executed.
20230328193125	[INFO]	FolderCreate: C:\Users\FerryvanGelderden\AppData\Roaming\HPOMEN16
20230328193125	[INFO]	FolderCreate successfully executed.
20230328193125	[INFO]	FileAppend write text 'Another line' to file: C:\Users\FerryvanGelderden\AppData\Roaming\test.txt (CP0)
20230328193125	[INFO]	FileAppend successfully executed.
20230328193125	[INFO]	FileAppend write text 'C:\Users\FerryvanGelderden\Documents\Tools\MSIX Helper' to file: C:\Users\FerryvanGelderden\AppData\Roaming\test.txt (CP0)
20230328193125	[INFO]	FileAppend successfully executed.
20230328193125	[INFO]	StringReplace text 'C:\Users\FerryvanGelderden\Documents\Tools\MSIX Helper' with 'C:\Users\FERRYV~1\AppData\Local\Temp' using source file C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64_cqqa1hc8w2dp\test.txt' for destination file: 'C:\Users\FerryvanGelderden\AppData\Roaming\test.txt' (OverWrite=1)
20230328193125	[INFO]	StringReplace successfully executed.
20230328193125	[INFO]	Download 'https://www.autohotkey.com/download/2.0/version.txt' for file: C:\Users\FerryvanGelderden\AppData\Local\version.txt
20230328193126	[INFO]	Download successfully executed.
20230328193126	[INFO]	RegWrite create 'REG_SZ' registry item 'WorkingDir' in 'HKEY_CURRENT_USER\Software\MyApp' with value 'C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64_cqqa1hc8w2dp'
20230328193126	[INFO]	RegWrite successfully executed.
20230328193126	[INFO]	Run and wait for Target: Notepad.exe " C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64_cqqa1hc8w2dp\Document.txt"
20230328193126	[INFO]	Working directory: C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64_cqqa1hc8w2dp
20230328193128	[INFO]	Run target with Process Id '20700' successfully stopped with return code: 0
20230328193128	[INFO]	[Executing target section: notepad]
20230328193128	[INFO]	Run and wait for Target: cmd.exe notepad
20230328193128	[INFO]	Working directory: C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64_cqqa1hc8w2dp
20230328193131	[INFO]	Target with Process Id '7140' stopped with return code: -1073741510
20230328193131	[INFO]	[Performing PostExit script section: PostExitScript]
20230328193131	[INFO]	FileDelete: C:\Users\FerryvanGelderden\AppData\Roaming\Document.txt
20230328193131	[INFO]	FileDelete successfully executed.
20230328193131	[INFO]	FolderDelete: C:\Users\FerryvanGelderden\Documents\Resource
20230328193131	[INFO]	FolderDelete successfully executed.
20230328193131	[INFO]	FolderDelete: C:\Users\FerryvanGelderden\AppData\Roaming\HPOMEN16
20230328193131	[INFO]	FolderDelete successfully executed.
20230328193131	[INFO]	FileDelete: C:\Users\FerryvanGelderden\AppData\Roaming\test.txt
20230328193131	[INFO]	FileDelete successfully executed.
20230328193131	[INFO]	FileDelete: C:\Users\FerryvanGelderden\AppData\Local\version.txt
20230328193131	[INFO]	FileDelete successfully executed.
20230328193131	[INFO]	RegDelete registry value 'WorkingDir' from 'HKEY_CURRENT_USER\Software\MyApp'
20230328193131	[INFO]	RegDelete successfully executed.
20230328193131	[INFO]	RegDelete registry key: HKEY_CURRENT_USER\Software\MyApp
20230328193131	[INFO]	RegDelete successfully executed.
20230328193131	[INFO]	MSIX Helper 1.0 stopped successfully.
-----	-----	-----





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.

The screenshot shows the 'MSIX Packaging Tool' window with the 'Create new package' tab selected. On the left is a sidebar with navigation links: 'Select environment', 'Prepare computer', 'Select installer', 'Package information' (selected), 'Installation', 'First launch tasks', 'Package report', and 'Create package'. The main area is titled 'Create new package' and contains the following fields:

- Package name ***: Text box containing 'MyPackage'.
- Package display name ***: Text box containing 'My Package, My-package1'.
- Publisher name ***: Text box containing 'CN=ESD'. To its right is the text 'Subject of the certificate provided'.
- Publisher display name ***: Text box containing 'My corporation'.
- Version ***: Four text boxes for version components, each containing '0'.
- Package Description**: Text box containing 'My Description'.
- Installation location**: Text box that is empty and highlighted with a red border. To its right is a 'Browse...' button.

Below these fields is a checkbox labeled 'Add support for MSIX Core to this package. See <https://aka.ms/MSIXCore> to learn more.' followed by the text 'Once you continue to the next page, installation will begin and you will not be able to return.' At the bottom right are three buttons: 'Previous', 'Next', and '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 Helper is to place the MSIX Helper executable with the corresponding INI file in a specified folder (For example "C:\MyApp") and set the "installation location" during the MSIX capturing process to this folder to ensure the MSIX Helper executable and the corresponding INI file will become available in the MSIX Package root folder when finishing the capture process. Optionally, you can create a shortcut from the Start Menu pointing to the MSIX Helper executable in the specified ("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 specified ("C:\MyApp") folder for using the PVAD location.





Developer information

My name is Ferry van Gelderen and I am an IT Professional that has more than 25 years of experience in the Microsoft application packaging/virtualization and deployment field. In 2014 I started the company Provolve IT with my companion Dereck Breuning and we started with the release of the "Easy Software Deployment" application that I started to develop some years prior. Today we are developing the "Easy Software Deployment Cloud" version for support on the Windows Azure platform. Developing tools to make our live easier is always a fun thing to do especially when you know this will help others move forward in there IT quest.

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

This software is created in AutoHotKey v2.0 and the source code is available for review and/or adjustments.

Learn more about Easy Software Deployment

<https://easysoftwaredeployment.nl/>

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

Website: <https://provolve.nl/>

Copyright © 2023 Provolve IT B.V.

