# MSIX Helper 1.2

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

## How to use

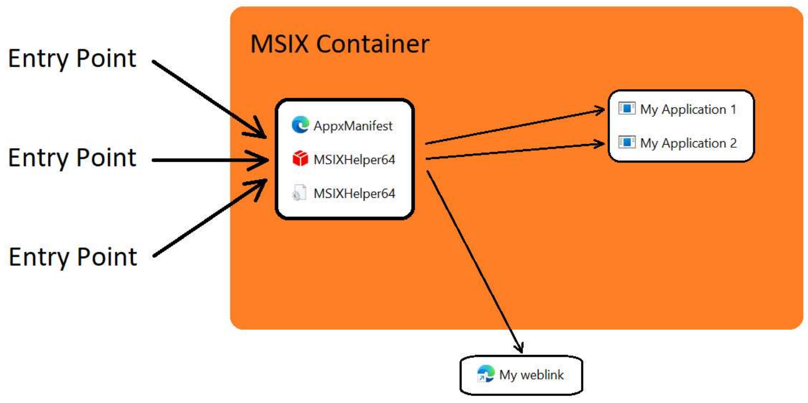
The MSIX Helper is a standalone single executable available in a 32-bit and 64-bit release. Use the 32-bit release for creating a 32-bit MSIX package developed on a 32-bit operating system that can be deployed on a 32-bit and/or 64-bit Windows 10/11 operating system and the 64-bit release for creating a 64-bit package developed on a 64-bit operating system 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.

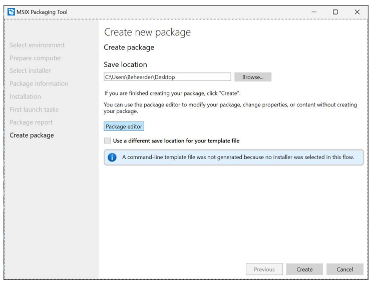
MSIXHelper64.exe
MSIXHelper32.exe
MSIXHelper64.ini
MSIXHelper32.ini

## Implementing the MSIX Helper

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 an MSIX Package copy the MSIX Helper executable and place it with the corresponding INI file in the MSIX package (root or sub) folder. 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 implementing the MSIX Helper, edit the **AppxManifest.xml** and/or **INI-file** during (or after) the MSIX Packaging capture process using the Microsoft MSIX Packaging Tool. When the MSIX capture process is finished, select the **Package editor** button to open the MSIX package.



Create new package

Create package

Save location

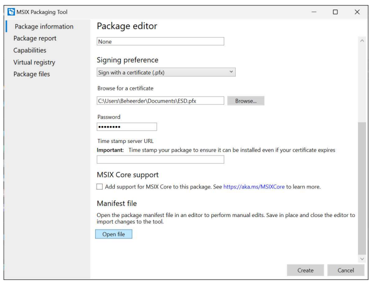
C:\Users\Beheerder\Desktop

If you are finished creating your package, click "Create".

You can use the package editor to modify your package, change properties, or content without creating your package.

Use a different save location for your template file

A command-line template file was not generated because no installer was selected in this flow.



Package editor

None

Signing preference

Sign with a certificate (.pfx)

Browse for a certificate

C:\Users\Beheerder\Documents\ESD.pfx

Password

••••••••

Time stamp server URL

**Important:** Time stamp your package to ensure it can be installed even if your certificate expires

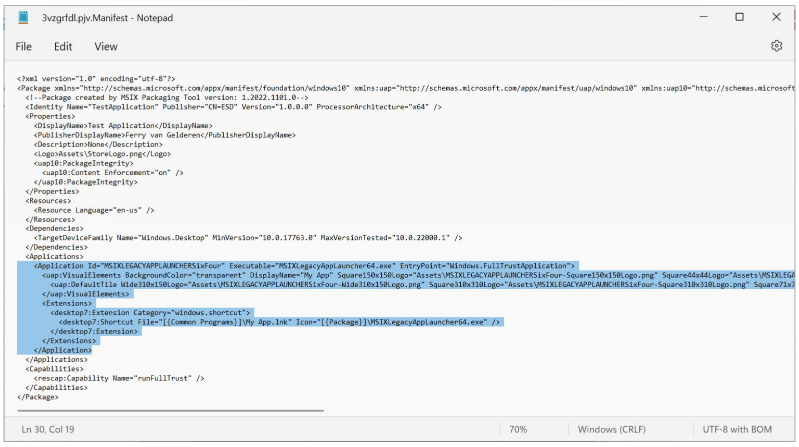
MSIX Core support

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

Manifest file

Open the package manifest file in an editor to perform manual edits. Save in place and close the editor to import changes to the tool.

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



```xml
<?xml version="1.0" encoding="utf-8"?>
<Package xmlns="http://schemas.microsoft.com/appx/manifest/foundation/windows10" xmlns:uap="http://schemas.microsoft.com/appx/manifest/uap/windows10" xmlns:uap10="http://schemas.microsoft
  <!--Package created by MSIX Packaging Tool version: 1.2022.1101.0-->
  <Identity Name="TestApplication" Publisher="CN=ESD" Version="1.0.0.0" ProcessorArchitecture="x64" />
  <Properties>
    <DisplayName>Test Application</DisplayName>
    <PublisherDisplayName>Ferry van Gelderen</PublisherDisplayName>
    <Description>None</Description>
    <Logo>Assets\StoreLogo.png</Logo>
    <uap10:PackageIntegrity>
      <uap10:Content Enforcement="on" />
    </uap10:PackageIntegrity>
  </Properties>
  <Resources>
    <Resource Language="en-us" />
  </Resources>
  <Dependencies>
    <TargetDeviceFamily Name="Windows.Desktop" MinVersion="10.0.17763.0" MaxVersionTested="10.0.22000.1" />
  </Dependencies>
  <Applications>
    <Application Id="MSIXLEGACYAPPLAUNCHERSixFour" Executable="MSIXLegacyAppLauncher64.exe" EntryPoint="Windows.FullTrustApplication">
      <uap:VisualElements BackgroundColor="transparent" DisplayName="My App" Square150x150Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square150x150Logo.png" Square44x44Logo="Assets\MSIXLEGA
        <uap:DefaultTile Wide310x150Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Wide310x150Logo.png" Square310x310Logo="Assets\MSIXLEGACYAPPLAUNCHERSixFour-Square310x310Logo.png" Square71x7
      </uap:VisualElements>
      <Extensions>
        <desktop7:Extension Category="windows.shortcut">
          <desktop7:Shortcut File="[{Common Programs}]\My App.lnk" Icon="[{Package}]\MSIXLegacyAppLauncher64.exe" />
        </desktop7:Extension>
      </Extensions>
    </Application>
  </Applications>
  <Capabilities>
    <rescap:Capability Name="runFullTrust" />
  </Capabilities>
</Package>
```

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

If multiple entry points are used, make sure that each entry point will have its own unique ***<Application Id=*** name and ***DisplayName=*** visual element name. When using the MSIX Helper for executing your entry point targets, make sure the entry points are starting the MSIX Helper executable (MSIXHelper32.exe or MSIXHelper64.exe) by checking the ***Executable=*** value.

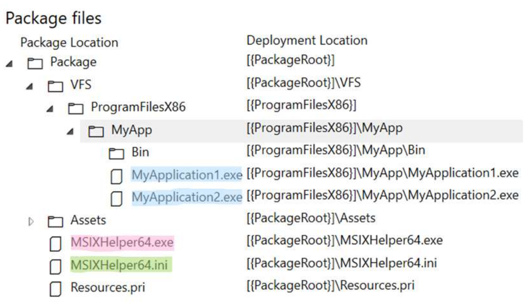
Change each ***<Application Id=*** name to the corresponding name of the application section that is defined in the MSIX Helper INI file (or vice versa) so the MSIX Helper executable will know what target (executable) to start when it is launched using the specified entry point.

**Keep in mind that for the AppxManifest.xml file the Application Id value can contain <u>ONLY ALPHA-NUMERIC</u> 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 MSIX Package Files:*



Package files

| Package Location | Deployment Location |
|---|---|
| Package | [{PackageRoot}] |
| VFS | [{PackageRoot}]\VFS |
| ProgramFilesX86 | [{ProgramFilesX86}] |
| MyApp | [{ProgramFilesX86}]\MyApp |
| Bin | [{ProgramFilesX86}]\MyApp\Bin |
| MyApplication1.exe | [{ProgramFilesX86}]\MyApp\MyApplication1.exe |
| MyApplication2.exe | [{ProgramFilesX86}]\MyApp\MyApplication2.exe |
| Assets | [{PackageRoot}]\Assets |
| MSIXHelper64.exe | [{PackageRoot}]\MSIXHelper64.exe |
| MSIXHelper64.ini | [{PackageRoot}]\MSIXHelper64.ini |
| Resources.pri | [{PackageRoot}]\Resources.pri |

*Example of the corresponding MSIX Helper INI file:*

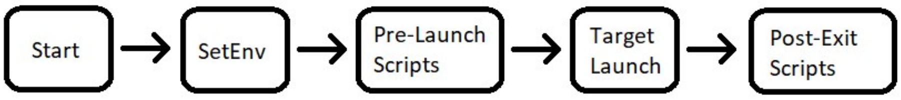
```
[ApplicationOne]
Target=%ProgramFiles(x86)%\MyApp\MyApplication1.exe
Options=
WorkingDir=%ProgramFiles(x86)%\MyApp
SetEnv=Environment
PreLaunch=PreLaunchScript

[ApplicationTwo]
Target=%ProgramFiles(x86)%\MyApp\MyApplication2.exe
Param1=-h
WorkingDir=%ProgramFiles(x86)%\MyApp
SetEnv=Environment
PreLaunch=PreLaunchScript

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

[Environment]
PATH=%ProgramFiles(x86)%\MyApp;%PATH%
AppBIN=%ProgramFiles(x86)%\MyApp\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%
```

## 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 can use the following key parameters:

**Target**
This is the target (executable) name to the (external) (legacy) program. For example: %PackagePath%\Bin\MyApplication.exe using PVAD or %ProgramFiles%\MyApp\Bin\MyApplication.exe using VFS for target executables (See Using PVAD or VFS). It is also possible to 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*). Do not use Quotes (“).
**WorkingDir**
This is the working directory for the launched target item. For example: %PackagePath%\Bin using PVAD or %ProgramFiles%\MyApp\Bin using VFS (See Using PVAD or VFS). Do not use Quotes (“).
**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 started from the Start Menu. Like the **WorkingDir** parameter this can also contain relative directory paths. For example: %WorkingDir%\data\test.cfg
Do not use Quotes (“). Quotes will be added automatically before and after the parameter value on execution if spaces are detected in the parameter value.
**Param2**
The second parameter to be used with the target executable started from the Start Menu. 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 from the Start Menu.
**Arg1**
The first parameter to be used when the target executable is started with an argument (MSIX Filetype Association). All assigned parameters (Arg1, Arg2, etc.) will be added before the MSIX Filetype Association arguments that will be added last. Do not use Quotes (“). Quotes will be added automatically before and after the parameter value on execution if spaces are detected in the parameter value.
**Arg2**
The second parameter to be used when the target executable is started with an argument (Filetype Association). You can continue adding more *Arg* options like Arg3, Arg4, etc. All *Arg* property values will be joined together with a space in between when the **Target** executable is executed. The MSIX Filetype Association arguments will be added automatically after the joined argument property values.
**SetEnv**
The name of the environment variable section that will be executed before performing the *PreLaunch* script section(s) 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.

Only the **Target** key parameter is required in the target section. The other key parameters are optional. The execution order for each Application Id target by the MSIX Helper process is always by executing the environment variables section **SetEnv** first, followed by the **PreLaunch** script section(s), then the **target** (executable) section and lastly the **PostExit** script section(s). If the SetEnv, PreLaunch or PostExit key parameter sections are not defined in the target section, the MSIX Helper process will skip these automatically and will only launch the target that is set in the target section.



*Example:*

```
[ApplicationOne]
Target=%PackagePath%\MyApp\MyApplication1.exe
Options=
WorkingDir=%PackagePath%\MyApp
Arg1=-OpenFile
SetEnv=Environment
PreLaunch=PreLaunchScript
PostExit=PostExitScript

[ApplicationTwo]
Target=%PackagePath%\MyApp\MyApplication2.exe
Options=Max
Param1=-Start
WorkingDir=%PackagePath%\MyApp
SetEnv=Environment
PreLaunch=PreLaunchScript
PostExit=PostExitScript

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

[Environment]
PATH=%PackagePath%\MyApp;%PATH%

[PreLaunchScript]
RegWrite, REG_SZ, HKEY_CURRENT_USER\Software\MyApp, WorkingDir, %PackagePath%\MyApp

[PostExitScript]
RegDelete, HKEY_CURRENT_USER\Software\MyApp, WorkingDir
```

## 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 executable 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__cqqwa1hc8w2dp*
**PackageFamilyName**
This will contain the family name of the MSIX package where the MSIX Helper is started from.
*Example: MSIXHelper64_cqqwa1hc8w2dp*
**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. These session environment variables can contain relative paths like the %WorkingDir% or %PackagePath% (for PVAD) or %ProgramFiles% (for VFS). These session 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=%PackagePath%;%PATH%
AppBIN=%PackagePath%\Bin
```

All environment variables that are defined under this section will be set starting at the top until the end of the section is reached. You can create more environment variable sections for other targets that need other session environment variables at startup or use the **EnvSet** script item in the PreLaunch and PostExit script sections to add or delete individual session environment variables. 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, EnvSet, RegRead, RegWrite, RegDelete, IniRead, IniWrite, IniDelete, Run, StringReplace, FileAppend** and **Download**.

Start each line under the *PreLaunch* and/or *PostExit* section name with the script item name and a comma (,) followed by 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.

| 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)*<br>0 | | |
| FolderCopy | Source Folder | Destination path | 1 *(overwrites)*<br>0 | | |
| FolderCreate | Folder name path | | | | |
| FileDelete | File(s) (wildcard) Pattern | | | | |
| FolderDelete | Folder name path<br><br>*(Deletes all files and subfolder)* | | | | |
| EnvSet | Name of the environment variable | Value to set the environment variable to. If omitted, the environment variable is deleted. | | | |
| RegRead | HKEY_LOCAL_MACHINE<br>HKEY_CURRENT_USER | Value name<br><br>(Value name and value will be set as an Environment Variable) | Default Value | | |
| RegWrite | REG_SZ<br>REG_EXPAND_SZ<br>REG_MULTI_SZ<br>REG_DWORD<br>REG_BINARY | HKEY_LOCAL_MACHINE<br>HKEY_CURRENT_USER | Value name | Value<br><br>*(If the value doesn't exist the value will be created)* | |
| RegDelete | HKEY_LOCAL_MACHINE<br>HKEY_CURRENT_USER | Value name<br><br>*(optional)* | | | |
| IniRead | INI Filename | Section name | Key name<br><br>(Key name and value will be set as an Environment Variable) | Default Value | |
| IniWrite | Value | INI Filename | Section name | Key name | |
| IniDelete | INI Filename | Section name | Key name<br><br>(optional) | | |
| Run | Target | WorkingDir | Min, Max or Hide<br><br>*(If omitted, the target executable launches normally)* | 0 *(wait for target command to end)*<br>1 *(the target command will be started without waiting for it to end)* | |
| StringReplace | Source Filename<br><br>*(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 or variable value to replace with<br>*(All occurrences of the search text will be replaced)* | Destination filename<br><br>*(If the file doesn't exist the file will be created first)* | 1 *(overwrites)*<br>0 *(Skip StringReplace when destination file already exists)* |
| FileAppend | Text write to the end of a file | Full file name path<br><br>*(If the file doesn't exist the file will be created first)* | UTF-8, UTF-8-RAW, UTF-16 or UTF-16-RAW<br><br>*(If omitted the default encoding will be ANSI)* | | |
| Download | URL (https/ftp) | Full file name path<br><br>*(Any existing file will be overwritten)* | | | |

*Environment Variables (%) can be used instead of fixed texts for file (paths) and registry values.*

Use the following If/Else statement items to execute a separate script section that can be executed on a specific value of an existing environment variable or whether a specified file or folder exist (or not).

**If, IfNot, IfExist, IfNotExist**

Start a line in the *PreLaunch* and/or *PostExit* section name with the statement item name and a comma (,) followed by a maximum of four item values also separated by a comma. The number of item values and what item values that are used are depending on the statement item used as described below.

| Script Item | Item Value 1 | Item Value 2 | Item Value 3 | Item Value 4 |
|---|---|---|---|---|
| If | The name of the environment variable value | The check value for the environment variable value | The name of the script section that needs to be executed when the value of the environment variable name is equal to the check value | The name of the script section that needs to be executed when the value of the environment variable name is **not** equal to the check value<br><br>(This item is optional) |
| IfNot | The name of the environment variable value | The check value for the environment variable value | The name of the script section that needs to be executed when the value of the environment variable name is **not** equal to the check value | The name of the script section that needs to be executed when the value of the environment variable name is equal to the check value<br><br>(This item is optional) |
| IfExist | File or Folder name path | The name of the script section that needs to be executed when the file or folder name path exists | The name of the script section that needs to be executed when the file or folder name path do **not** exists<br><br>(This item is optional) | |
| IfNotExist | File or Folder name path | The name of the script section that needs to be executed when the file or folder name path do **not** exists | The name of the script section that needs to be executed when the file or folder name path exists<br><br>(This item is optional) | |

*Use the script items RegRead and/or IniRead to get a specified registry value or key value from an .ini file that will be set as an environment variable for the If and IfNot statement items to execute a separate script section depending on the registry or INI file key item value.*

All script items that are defined under the script sections will be set starting at the top until the end of the section is reached.

## Explanation Script Items

We will explain each script item in detail on how to use them.

### *FileCopy*

Copies one or more files.

- Source pattern

  The name of a single file or folder, or a wildcard pattern such as C:\Temp\*.tmp. SourcePattern is assumed to be in %WorkingDir% if an absolute path isn't specified.

- Destination pattern

  The name or pattern of the destination, which is assumed to be in %WorkingDir% (not writable by default) if an absolute path isn't specified. The destination directory must already exist. Environment variables can be used like %APPDATA%\MyDir.

- Overwrite

  This parameter determines whether to overwrite files if they already exist. If this parameter is 1 (true), the command overwrites existing files. If omitted or 0 (false), the command does not overwrite existing files.

## *FolderCopy*

Copies a folder along with all its sub-folders and files (similar to xcopy).

- Source

  Name of the source directory (with no trailing backslash), which is assumed to be in %WorkingDir% if an absolute path isn't specified. For example: C:\My Folder

- Destination

  Name of the destination directory (with no trailing baskslash), which is assumed to be in %WorkingDir% (not writable by default) if an absolute path isn't specified. Environment variables can be used.

- Overwrite

  This parameter determines whether to overwrite files if they already exist. Specify one of the following values:
  0 (false): Do not overwrite existing files. The operation will fail and have no effect if Dest already exists as a file or directory.
  1 (true): Overwrite existing files. However, any files or subfolders inside Dest that do not have a counterpart in Source will not be deleted.

## *FolderCreate*

Creates a folder.

- DirName

  Name of the directory to create, which is assumed to be in %WorkingDir% (not writable by default) if an absolute path isn't specified. Environment variables can be used like %LOCALAPPDATA%\DirName

## *FileDelete*

Deletes one or more files.

- FilePattern

  The name of a single file or a wildcard pattern such as C:\Temp\*.tmp. FilePattern is assumed to be in %WorkingDir% (not writable by default) if an absolute path isn't specified.

## *FolderDelete*

Deletes a folder.

- DirName

  Name of the directory to delete, which is assumed to be in %WorkingDir%  (not writable by default) if an absolute path isn't specified. This will remove all files and subdirectories (like "rmdir /S").

## *EnvSet*

Writes a value to a variable contained in the environment or deletes the environment variable.

- EnvVar

  Name of the environment variable to use, e.g. "COMSPEC" or "PATH".
- Value

  Value to set the environment variable to. If omitted, the environment variable is deleted.

No permanent environment variables will be set for the current user or system.

### *RegRead*

Reads a value from the registry and set the value name and value as an environment variable.

- KeyName

  The full name of the registry key. This must start with HKEY_LOCAL_MACHINE, HKEY_USERS, HKEY_CURRENT_USER, HKEY_CLASSES_ROOT, or HKEY_CURRENT_CONFIG (or the abbreviations for each of these, such as HKLM).

- ValueName

  The name of the value to retrieve. Together with the value itself this will be set as an environment variable that can be used for other PreLaunch or PostExit script actions or launching the Target executable.

- DefaultValue

  The default value to return if the specified key or value does not exist.

### *RegWrite*

Writes a value to the registry.

- ValueType

  Must be either REG_SZ, REG_EXPAND_SZ, REG_MULTI_SZ, REG_DWORD, or REG_BINARY.

- KeyName

  The full name of the registry key. This must start with HKEY_LOCAL_MACHINE, HKEY_USERS, HKEY_CURRENT_USER, HKEY_CLASSES_ROOT, or HKEY_CURRENT_CONFIG (or the abbreviations for each of these, such as HKCU).

- ValueName

  The name of the value that will be written to. If blank or omitted, the key's default value will be used. The default value is displayed as "(Default)" by RegEdit.

- Value

  The value to be written. This can also be an environment variable like %PackagePath% where the actual value will be used.

### *RegDelete*

Deletes a value from the registry.

- KeyName

  The full name of the registry key. This must start with HKEY_LOCAL_MACHINE, HKEY_USERS, HKEY_CURRENT_USER, HKEY_CLASSES_ROOT, or HKEY_CURRENT_CONFIG (or the abbreviations for each of these, such as HKLM).

- ValueName

  The name of the value to delete. If blank or omitted, the registry subkey will be deleted.

### *IniRead*

Reads a value from a standard format .ini file and set the key name and value as an environment variable.

- Filename

  The name of the .ini file, which is assumed to be in %WorkingDir% if an absolute path isn't specified.

- Section

  The section name in the .ini file, which is the heading phrase that appears in square brackets (do not include the brackets in this parameter).

- Key

  The key name in the .ini file.

- Default

  If omitted, an Error is thrown on failure. Otherwise, specify the value to return on failure.

### *IniWrite*

Writes a value or section to a standard format .ini file.

- Value

  The string or number that will be written to the right of Key's equal sign (=).

- Filename

  The name of the .ini file, which is assumed to be in %WorkingDir% if an absolute path isn't specified.

- Section

  The section name in the .ini file, which is the heading phrase that appears in square brackets (do not include the brackets in this parameter).

- Key

  The key name in the .ini file.

### *IniDelete*

Deletes a value from a standard format .ini file.

- Filename

  The name of the .ini file, which is assumed to be in %WorkingDir% if an absolute path isn't specified.

- Section

  The section name in the .ini file, which is the heading phrase that appears in square brackets (do not include the brackets in this parameter).

- Key

  If omitted, the entire section will be deleted. Otherwise, specify the key name in the .ini file.

Provolve IT

### *Run*

Runs an external program.

- Target

  A document, URL, executable file (.exe, .com, .bat, etc.), shortcut (.lnk), or system verb to launch. If Target is a local file and no path was specified with it, %WorkingDir% will be searched first. If no matching file is found there, the system will search for and launch the file if it is integrated ("known"), e.g. by being contained in one of the PATH folders. To pass parameters, add them immediately after the program or document name.

- WorkingDir

  The working directory for the launched item. Do not enclose the name in double quotes even if it contains spaces. If omitted, the own working directory (%WorkingDir%) will be used.

- Options

  If omitted, the function launches Target normally. To change this behavior, specify one or more of the following words:
  Max: launch maximized
  Min: launch minimized
  Hide: launch hidden (cannot be used in combination with either of the above).

- NoWait

  This parameter determines whether to wait for the external program to end or immediately continue after the external program is started. Specify one of the following values:
  0 (false): Wait for the external program to end before continuing.
  1 (true): Immediately continue after the external program is started.

### *StringReplace*

Replaces the specified substring with a new string in a file.

- Source filename

  The full path to the file name that contains the file content with the specified substring that needs to be replaced. For example: %PackagePath%\Config.cfg

- Search text

  The substring value to search for that needs to be replaced. The file content can contain one or multiple occurrences that needs to be replaced. For example: <AppData>

- Replace value

  The new value text to replace the search text with. All occurrences of the search text will be replaced. Environment variables can be used and will be replaced with the actual value. For example: %APPDATA%

- Destination filename

  The full path to the (non-existing) file path that needs to contain the new replaced file content. If the destination file doesn't exist the file will be created first. If the source and destination file names are equal, multiple string replacements with different search texts and replacement values can be performed for the same file and the overwrite option will be ignored. For example: %LOCALAPPDATA%\MyApp\Config.cfg

- Overwrite

  This parameter determines whether to overwrite the destination file if it already exist. Specify one of the following values:
  0 (false): Do not overwrite the existing destination file. The operation will be ignored when the destination file already exists.
  1 (true): Overwrite the existing destination file. Even if the file already exists, the operation will be performed.

### *FileAppend*

Writes text or binary data to the end of a file (first creating the file, if necessary).

- Text

  The text or raw binary data to append to the file. A linefeed characters (`n) will be added after each text or binary data added. Make sure a linefeed exists after the existing text or binary content to make sure the new text or binary data is added as a new line.

- Filename

  The name of the file to be appended, which is assumed to be in %WorkingDir% if an absolute path isn't specified. The destination directory must already exist.

- Options

  Encoding: Specify any of the encoding names UTF-8, UTF-8-RAW, UTF-16 or UTF-16-RAW. If no encoding name is specified the default existing coding will be used.

### Download

Downloads a file from the Internet.

- URL

  URL of the file to download. For example, http://someorg.org might retrieve the welcome page for that organization.

- Filename

  Specify the name of the file to be created locally, which is assumed to be in %A_WorkingDir% if an absolute path isn't specified. Any existing file will be overwritten by the new file.

**Keep in mind that writing to the MSIX packaging folder or HKEY_LOCAL_MACHINE registry is not allowed.**
All script items that are defined under the script section will be set starting at the top until the end of the script section is reached. 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
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%
IniWrite, %TEMP%, %USERPROFILE%\Documents\document.ini, Main, TempFolder
Iniwrite, Production, %USERPROFILE%\Documents\document.ini, Main, Environment
IniRead, %USERPROFILE%\Documents\document.ini, Main, Environment
IniDelete, %USERPROFILE%\Documents\document.ini, Main, Environment
Run, Notepad.exe "%WorkingDir%\Document.txt", %WorkingDir%,, 0
RegRead, HKEY_CURRNT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings, ProxyEnable, 0
If, ProxyEnable, 1, SetProxyScript

[SetProxyScript]
RegWrite, REG_DWORD, HKEY_CURRENT_USER\Software\MyApp, ProxyEnable, 1

[PostExitScript]
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
IfExist, %APPDATA%\Document.txt, DeleteDocument

[DeleteDocument]
FileDelete, %APPDATA%\Document.txt
```

## 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

## Using the MSIX Helper in a non-MSIX environment

Although the MSIX Helper is primarily used for running within the MSIX packaging environment, it is possible to use the MSIX Helper in other packaging environments like App-V, ThinApp or MSI. For this to work you need to start the MSIX Helper executable with the target section name as a parameter that is defined in the MSIX Helper INI file. For example:

```
.\MSIXHelper64.exe TargetName
```

*MSIXHelper64.ini Example:*

```
[TargetName]
Target=%ComSpec%
```

The example above will run the command prompt using the MSIX Helper in a non-MSIX environment. The **AppId** and **PackageFamilyName** session environment variables will be set to the target section name and both the **WorkingDir** and **PackagePath** session environment variables will be set to the path location where the MSIX Helper executable is started. For troubleshooting, create an empty file with the target section name and the .log extension in the personal temp folder for log information when the MSIX Helper executable is executed. For example: %TEMP%\TargetName.log (See Troubleshooting for more information)

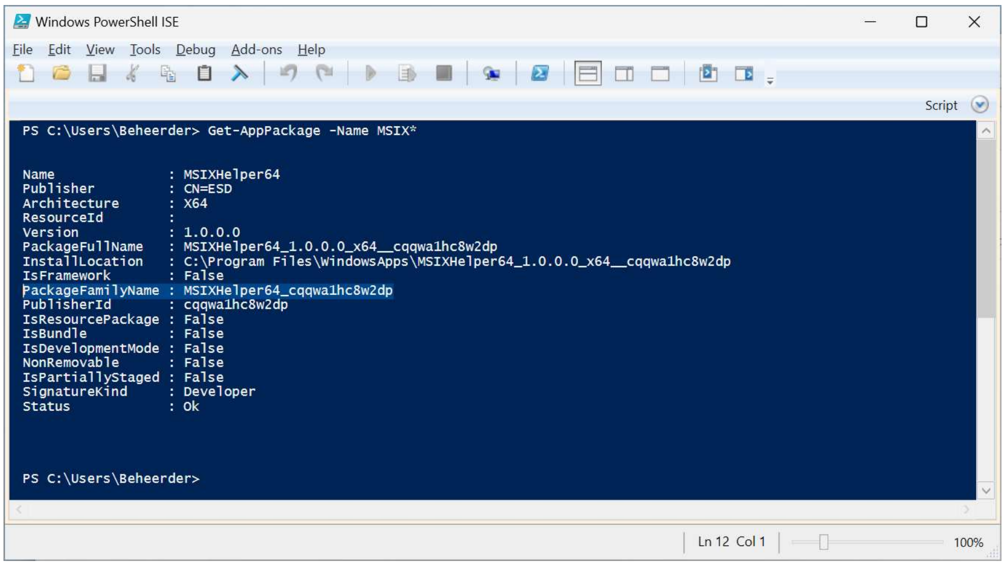
## Troubleshooting

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

If you want to use a log file for troubleshooting you 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_cqqwa1hc8w2dp.log

You can use the Powershell cmdlet *Get-AppPackage* to find the PackagFamilyName 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__cqqwa1hc8w2dp
InstallLocation   : C:\Program Files\WindowsApps\MSIXHelper64_1.0.0.0_x64__cqqwa1hc8w2dp
IsFramework       : False
PackageFamilyName : MSIXHelper64_cqqwa1hc8w2dp
PublisherId       : cqqwa1hc8w2dp
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.

It is also possible to use an external INI file for the MSIX Helper (inside the MSIX package) to use instead of the INI file that is already part of the existing MSIX package. This will give you the opportunity to test changes within the MSIX Helper INI file without changing the existing MSIX package. Like the log-file, manually create the INI file in the personal temp folder that has the PackageFamilyName property name and the .ini extension.

For example: %TEMP%\MSIXHelper64_cqqwa1hc8w2dp.ini

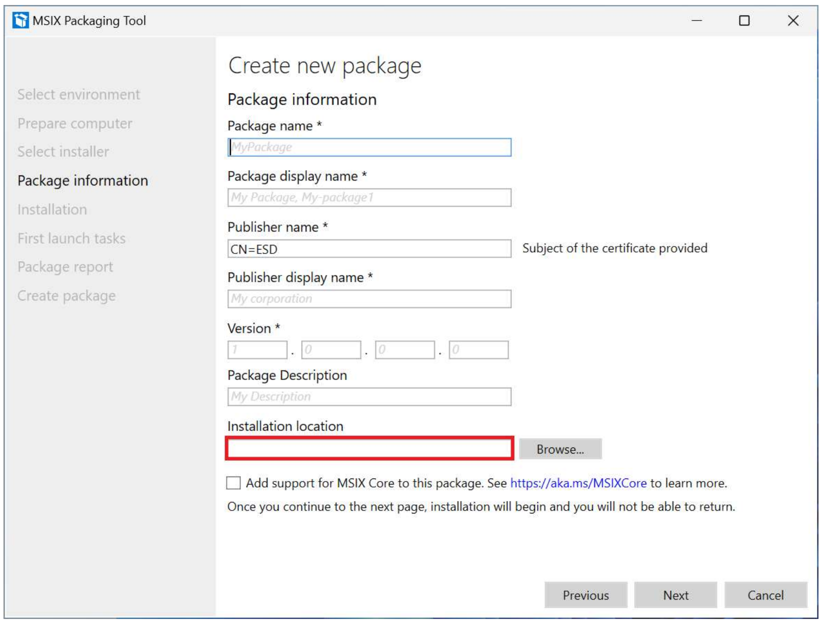
Keep in mind that the INI file created in the personal temp folder will always be used as long as this file exists. When removing the INI file from the personal temp folder the existing MSIX Helper INI file that is part of the MSIX package will be used again.

*Example logfile:*

| Timestamp | Level | Details |
|---|---|---|
| 20231015160510 | [ INFO ] | MSIX Helper 1.2 started for Application Id: KEEPASS |
| 20231015160510 | [ INFO ] | Computer name: WIN-10E-NL-X64 |
| 20231015160510 | [ INFO ] | Operating System version: 10.0.19045 (64-bit=1) |
| 20231015160510 | [ INFO ] | Operating System language code: 0413 |
| 20231015160510 | [ INFO ] | User name: Administrator |
| 20231015160510 | [ INFO ] | Working directory: C:\Program Files\WindowsApps\KeePassPasswordSafe_2.55.0.0_x86__cqqwa1hc8w2dp |
| 20231015160510 | [ INFO ] | MSIX PackagePath: C:\Program Files\WindowsApps\KeePassPasswordSafe_2.55.0.0_x86__cqqwa1hc8w2dp |
| 20231015160510 | [ INFO ] | MSIX PackageFamilyName: KeePassPasswordSafe_cqqwa1hc8w2dp |
| 20231015160510 | [ INFO ] | MSIX AppId: KEEPASS |
| 20231015160510 | [ INFO ] | INI file: C:\Program Files\WindowsApps\KeePassPasswordSafe_2.55.0.0_x86__cqqwa1hc8w2dp\MSIXHelper32.ini |
| 20231015160510 | [ INFO ] | [ Performing PreLaunch script section: PreLaunchScript ] |
| 20231015160510 | [ INFO ] | Using IfNotExist statement for file or folder 'C:\Users\Administrator\Programs'. When true goto script section: CreateProgramsFolder (Else goto script section: ) |
| 20231015160510 | [ INFO ] | [ Performing script section: CreateProgramsFolder ] |
| 20231015160510 | [ INFO ] | FolderCreate: C:\Users\Administrator\Programs |
| 20231015160510 | [ INFO ] | FolderCreate successfully executed. |
| 20231015160510 | [ INFO ] | Using IfNotExist statement for file or folder 'C:\Users\Administrator\Programs\KeePass Password Safe 2'. When true goto script section: CopyKeePass (Else goto script section: ) |
| 20231015160510 | [ INFO ] | [ Performing script section: CopyKeePass ] |
| 20231015160510 | [ INFO ] | FolderCopy from source 'C:\Program Files\WindowsApps\KeePassPasswordSafe_2.55.0.0_x86__cqqwa1hc8w2dp\KeePass Password Safe 2' to destination 'C:\Users\Administrator\Programs\KeePass Password Safe 2' (OverWrite=1) |
| 20231015160510 | [ INFO ] | FileCopy successfully executed. |
| 20231015160510 | [ INFO ] | [ Executing target section: KEEPASS ] |
| 20231015160510 | [ INFO ] | Run and wait for Target: C:\Users\Administrator\Programs\KeePass Password Safe 2\KeePass.exe |
| 20231015160510 | [ INFO ] | Working directory: C:\Users\Administrator\Programs\KeePass Password Safe 2 |
| 20231015160526 | [ INFO ] | Target with Process Id '5680' stopped with return code: 0 |
| 20231015160526 | [ INFO ] | [ Performing PostExit script section: PostExitScript ] |
| 20231015160526 | [ INFO ] | Using IfExist statement for file or folder 'C:\Users\Administrator\Programs\KeePass Password Safe 2'. When true goto script section: DeleteKeePass (Else goto script section: ) |
| 20231015160526 | [ INFO ] | [ Performing script section: DeleteKeePass ] |
| 20231015160526 | [ INFO ] | FolderDelete: C:\Users\Administrator\Programs\KeePass Password Safe 2 |
| 20231015160526 | [ INFO ] | FolderDelete successfully executed. |
| 20231015160526 | [ INFO ] | MSIX Helper 1.2 stopped successfully. |
| -------------- | --------- | ------------------------------------------------------------------------------------------------------------------------ |
| 20231015160541 | [ INFO ] | MSIX Helper 1.2 started for Application Id: KEEPASS |
| 20231015160541 | [ INFO ] | Computer name: WIN-10E-NL-X64 |
| 20231015160541 | [ INFO ] | Operating System version: 10.0.19045 (64-bit=1) |
| 20231015160541 | [ INFO ] | Operating System language code: 0413 |
| 20231015160541 | [ INFO ] | User name: Administrator |
| 20231015160541 | [ INFO ] | Working directory: C:\Program Files\WindowsApps\KeePassPasswordSafe_2.55.0.0_x86__cqqwa1hc8w2dp |
| 20231015160541 | [ INFO ] | MSIX PackagePath: C:\Program Files\WindowsApps\KeePassPasswordSafe_2.55.0.0_x86__cqqwa1hc8w2dp |
| 20231015160541 | [ INFO ] | MSIX PackageFamilyName: KeePassPasswordSafe_cqqwa1hc8w2dp |
| 20231015160541 | [ INFO ] | MSIX AppId: KEEPASS |
| 20231015160541 | [ INFO ] | INI file: C:\Program Files\WindowsApps\KeePassPasswordSafe_2.55.0.0_x86__cqqwa1hc8w2dp\MSIXHelper32.ini |
| 20231015160541 | [ INFO ] | [ Performing PreLaunch script section: PreLaunchScript ] |
| 20231015160541 | [ INFO ] | Using IfNotExist statement for file or folder 'C:\Users\Administrator\Programs'. When true goto script section: CreateProgramsFolder (Else goto script section: ) |
| 20231015160541 | [ INFO ] | Using IfNotExist statement for file or folder 'C:\Users\Administrator\Programs\KeePass Password Safe 2'. When true goto script section: CopyKeePass (Else goto script section: ) |
| 20231015160541 | [ INFO ] | [ Performing script section: CopyKeePass ] |
| 20231015160541 | [ INFO ] | FolderCopy from source 'C:\Program Files\WindowsApps\KeePassPasswordSafe_2.55.0.0_x86__cqqwa1hc8w2dp\KeePass Password Safe 2' to destination 'C:\Users\Administrator\Programs\KeePass Password Safe 2' (OverWrite=1) |
| 20231015160541 | [ INFO ] | FileCopy successfully executed. |
| 20231015160541 | [ INFO ] | [ Executing target section: KEEPASS ] |
| 20231015160541 | [ INFO ] | Run and wait for Target: C:\Users\Administrator\Programs\KeePass Password Safe 2\KeePass.exe "C:\Users\Administrator\Documents\My Database.kdbx" |
| 20231015160541 | [ INFO ] | Working directory: C:\Users\Administrator\Programs\KeePass Password Safe 2 |
| 20231015160641 | [ INFO ] | Target with Process Id '6516' stopped with return code: 0 |
| 20231015160641 | [ INFO ] | [ Performing PostExit script section: PostExitScript ] |
| 20231015160641 | [ INFO ] | Using IfExist statement for file or folder 'C:\Users\Administrator\Programs\KeePass Password Safe 2'. When true goto script section: DeleteKeePass (Else goto script section: ) |
| 20231015160641 | [ INFO ] | [ Performing script section: DeleteKeePass ] |
| 20231015160641 | [ INFO ] | FolderDelete: C:\Users\Administrator\Programs\KeePass Password Safe 2 |
| 20231015160641 | [ INFO ] | FolderDelete successfully executed. |
| 20231015160641 | [ INFO ] | MSIX Helper 1.2 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.



Create new package

Package information

Package name *: MyPackage

Package display name *: My Package, My-package1

Publisher name *: CN=ESD — Subject of the certificate provided

Publisher display name *: My corporation

Version *: 1 . 0 . 0 . 0

Package Description: My Description

Installation location:

☐ 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.

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, together with my companion Dereck Breuning, we started the company Provolve IT 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/