# **Zero-Config Executable Installation**

An extension for the PowerShell App Deployment Toolkit

Administration Guide

Document Version 1.0.0

09/11/2019

Developed by Kevin Street

<http://kevinstreet.co.uk>

Table of Contents

[Zero-Config Executable Installation - 1 -](#_Toc24218265)

[Overview - 4 -](#_Toc24218266)

[Introduction - 4 -](#_Toc24218267)

[Features - 4 -](#_Toc24218268)

[System Requirements - 4 -](#_Toc24218269)

[Licensing - 4 -](#_Toc24218270)

[Integration with PowerShell App Deployment Toolkit - 6 -](#_Toc24218271)

[Changes to make to Deploy-Application.ps1 - 6 -](#_Toc24218272)

[Changes to make to AppDeployToolkitExtensions.ps1 - 8 -](#_Toc24218273)

[Files to copy - 8 -](#_Toc24218274)

[Extension Information - 9 -](#_Toc24218275)

[Testing if your application is compatible - 9 -](#_Toc24218276)

[Using the extension - 10 -](#_Toc24218277)

[Additional requirements - 10 -](#_Toc24218278)

[Multiple .exe files in the Files directory - 10 -](#_Toc24218279)

[Inno Setup installations - 10 -](#_Toc24218280)

[InstallShield installations - 11 -](#_Toc24218281)

[Office 365 click-to-run installations - 11 -](#_Toc24218282)

[Logging - 11 -](#_Toc24218283)

[Digital Signature - 12 -](#_Toc24218284)

[Example Deployments - 12 -](#_Toc24218285)

[Deploying Notepad++ with ZeroConfigExe - 12 -](#_Toc24218286)

[Deploying Office 365 with ZeroConfigExe - 14 -](#_Toc24218287)

[Extension Variables - 16 -](#_Toc24218288)

[Extension Functions - 17 -](#_Toc24218289)

[Test-ExeOrMsuIsPresent - 17 -](#_Toc24218290)

[Start-Installation - 17 -](#_Toc24218291)

[Find-InstallerTechnology - 17 -](#_Toc24218292)

[Find-UninstallStringInRegsitry - 18 -](#_Toc24218293)

[Install-UsingWindowsInstaller - 18 -](#_Toc24218294)

[Install-UsingNSISInstaller - 19 -](#_Toc24218295)

[Install-UsingInnoSetupInstaller - 19 -](#_Toc24218296)

[Install-UsingInstallShieldInstaller - 20 -](#_Toc24218297)

[Install-UsingWixBurnInstaller - 20 -](#_Toc24218298)

[Install-UsingWiseInstaller - 21 -](#_Toc24218299)

[Install-UsingInstallAWARE - 22 -](#_Toc24218300)

[Install-UsingInstall4j - 22 -](#_Toc24218301)

[Install-UsingSetupFactory - 23 -](#_Toc24218302)

[Install-UsingOffice365ClickToRunInstaller - 23 -](#_Toc24218303)

[Install-UsingWindowsUpdateStandaloneInstaller - 24 -](#_Toc24218304)

[Test-SupportedInstallerType - 25 -](#_Toc24218305)

# Overview

## Introduction

The Zero-Config Executable (ZeroConfigExe) Installation script is an extension to the PowerShell App Deployment Toolkit designed to make deploying executables as simple as deploying MSI files. The PSADT allows an administrator to simply drop an MSI file into the Files directory and run Deploy-Application.exe file to automatically install the MSI with default silent parameters. With the addition of this extension the administrator is given the same ability with executable installers.

## Features

This extension supports the following installer technologies:

* Windows Installer
* NSIS
* Inno Setup
* InstallShield
* WiX Burn
* Wise
* InstallAWARE
* install4j
* Setup Factory

It also has special logic for installing the following products:

* Microsoft Office 365 click-to-run
* Microsoft Windows update files (.msu)

The script scans the installer file for a reference to one of the installer technologies and then proceeds to silently install it. Uninstalls are also supported and both installs and uninstalls are logged in the administrator defined logging location when supported by the installer.

This extension furthers the goal of the PowerShell App Deployment Toolkit by offering an easy to use, powerful and consistent deployment experience for both the administrator and end user.

## System Requirements

This extension has been tested on Windows 7, 8.1 and 10, and their server equivalents, on both x86 and x64 platforms. PowerShell version 2 is the minimum required version, which is installed by default on Windows 7 and above.

## Licensing

Zero-Config Executable Installation.

Copyright © 2019 - Kevin Street.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <http://www.gnu.org/licenses>.

# Integration with PowerShell App Deployment Toolkit

These steps need to be followed to make the ZeroConfigExe extension available in the PowerShell App Deployment Toolkit. This extension is designed so that it does not rely on any particular version of the PSADT and can be used with future versions. You will need to re-integrate it any time you upgrade the PSADT (if you replace Deploy-Application.ps1 and AppDeployToolkitExtensions.ps1).

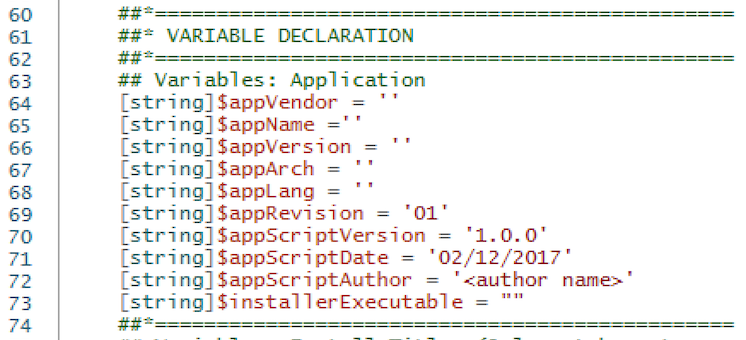
**A copy of Deploy-Application.ps1 and AppDeployToolkitExtensions.ps1 with these modifications is included in the download of ZeroConfigExe for your convenience.**

## Changes to make to Deploy-Application.ps1

Add the following variable in the VARIABLE DECLARATION section:

[string]$installerExecutable = ""

Example of how it should look:

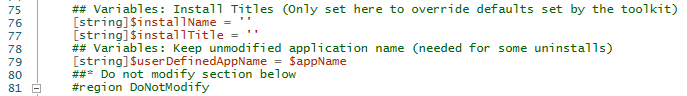


Add the following block of code in the VARIABLE DECLARATION section underneath the main block of variables:

## Variables: Keep unmodified application name (needed for some uninstalls)

[string]$userDefinedAppName = $appName

Example of how it should look:



Add the following block of code in the PRE-INSTALLATION section in-between Show-InstallationWelcome and Show-InstallationProgress:

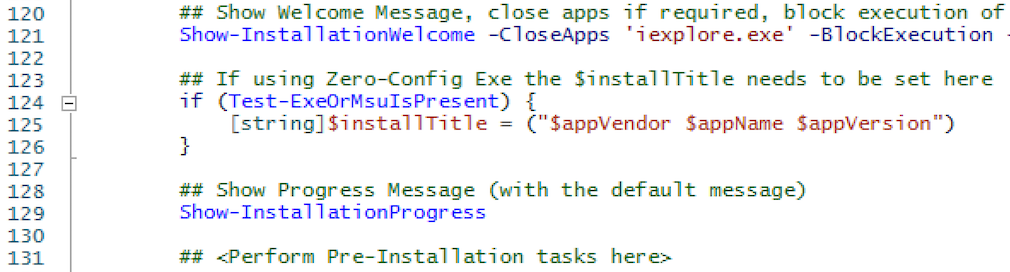
## If using Zero-Config Exe the $installTitle needs to be set here

if (Test-ExeOrMsuIsPresent) {

[string]$installTitle = ("$appVendor $appName $appVersion")

}

Example of how it should look:



Add the following block of code in the INSTALLATION section in-between the Execute-MSI and <Perform Installation tasks here> sections:

## Handle Zero-Config Exe Installation

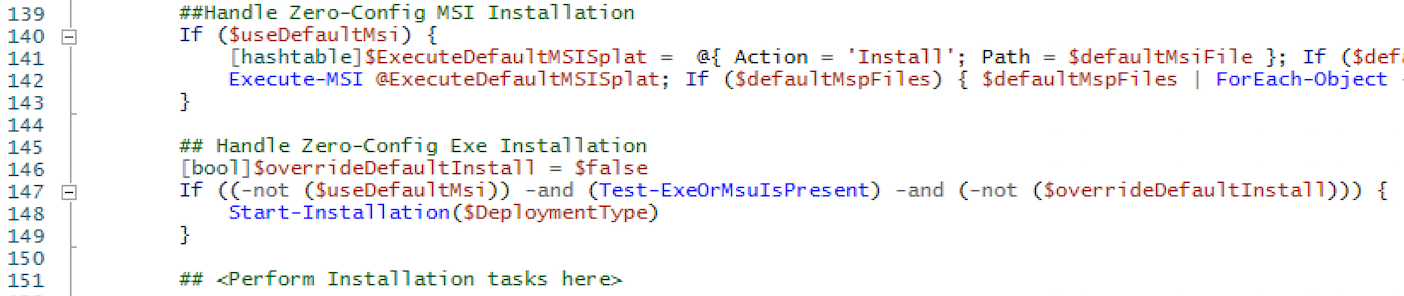
[bool]$overrideDefaultInstall = $false

If ((-not ($useDefaultMsi)) -and (Test-ExeOrMsuIsPresent) -and (-not ($overrideDefaultInstall))) {

Start-Installation($DeploymentType)

}

Example of how it should look:



Add the following block of code in the PRE-UNINSTALLATION section in-between Show-InstallationWelcome and Show-InstallationProgress:

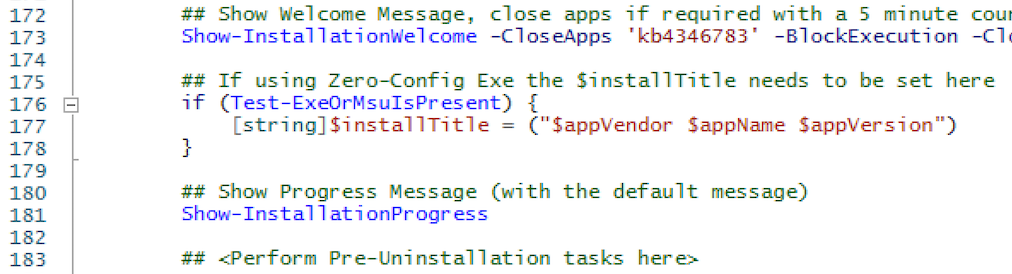
## If using Zero-Config Exe the $installTitle needs to be set here

if (Test-ExeOrMsuIsPresent) {

[string]$installTitle = ("$appVendor $appName $appVersion")

}

Example of how it should look:



Add the following block of code in the UNINSTALLATION section in-between the Execute-MSI and <Perform Uninstallation tasks here> sections:

## Handle Zero-Config Exe Uninstallation

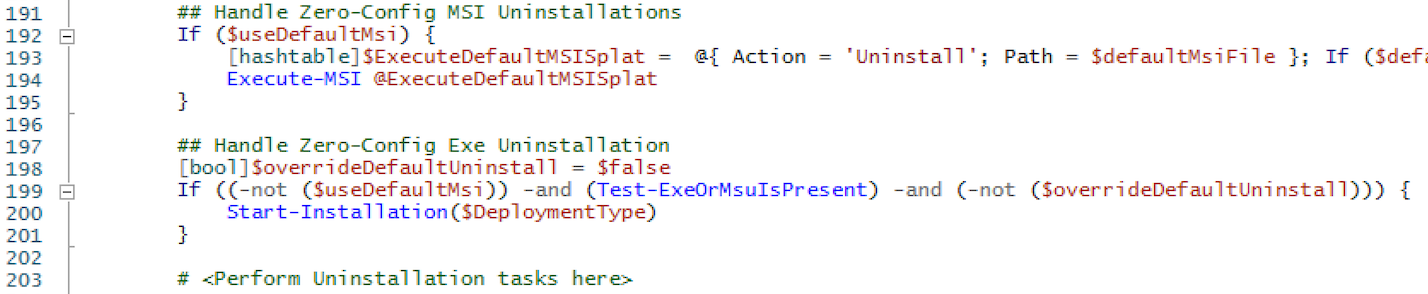
[bool]$overrideDefaultUninstall = $false

If ((-not ($useDefaultMsi)) -and (Test-ExeOrMsuIsPresent) -and (-not ($overrideDefaultUninstall))) {

Start-Installation($DeploymentType)

}

Example of how it should look:



## Changes to make to AppDeployToolkitExtensions.ps1

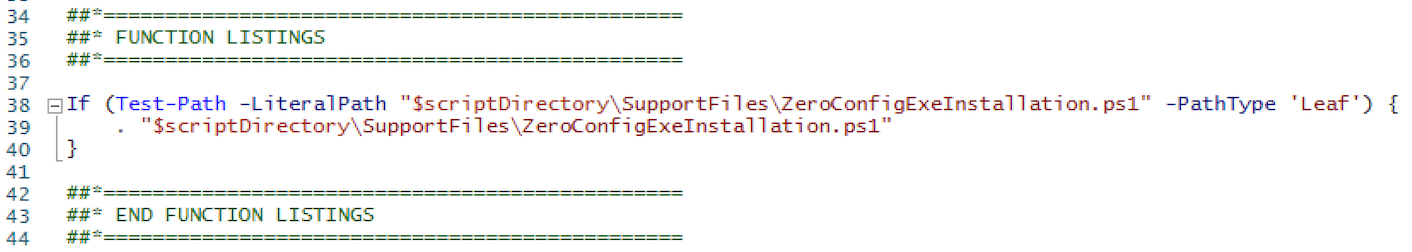
Add the following block of code in the FUNCTION LISTINGS section:

If (Test-Path -LiteralPath "$scriptDirectory\SupportFiles\ZeroConfigExeInstallation.ps1" -PathType 'Leaf') {

. "$scriptDirectory\SupportFiles\ZeroConfigExeInstallation.ps1"

}

Example of how it should look:



## Files to copy

Copy ZeroConfigExeInstallation.ps1 into the SupportFiles directory of the PSADT.

# Extension Information

This script is a best effort attempt to make installing executables as easy as installing MSIs. Although there are general guidelines about how each installer handles silent installation, the install process can be customised by the vendor and therefore there is no guarantee that this script will work for all software.

The simplest way to use the extension is to copy an installer file into the Files directory of the PSADT and update Deploy-Application.ps1 with the $appName variable and the -CloseApps parameter. The script will locate the vendor, name and version information to display during the installation and proceed to install it silently with no restart.

Some installers require more than just the installer file in order to install silently. InstallShield installers require an install and uninstall answer file and Office 365 installations require a configuration file. Inno Setup installers can accept an answer file if provided but will install with default parameters if none is supplied. More details on these cases can be found below, in the section [Additional requirements](#_Additional_requirements).

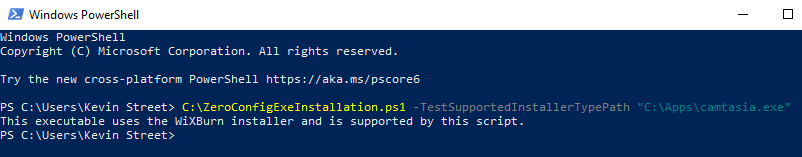
You do not need to modify ZeroConfigExeInstallation.ps1 in order to use it for your application deployments. See PSAppDeploymentToolkitAdminGuide.docx for instructions on using the PSADT as a whole.

## Testing if your application is compatible

Before you start you can test whether the application you wish to use with ZeroConfigExe is using an installer technology that is known and supported by the script. To do this, follow these steps:

1. Open a PowerShell window and navigate to the location the ZeroConfigExeInstallation.ps1 file is held.
2. Use the command “ZeroConfigExeInstallation.ps1 -TestSupportedInstallerTypePath “PathToInstallerExe”

For example:



This is not a guarantee that the application will install or uninstall successfully as some vendors modify their installers to make them incompatible with the documented silent install and uninstall switches that this script uses.

## Using the extension

The steps to use this extension are as follows:

1. Copy the installation file and supporting files (if any) into the Files directory of the PSADT.
2. Edit Deploy-Application.ps1 and make the following changes:
   1. Assign the variable [string]$appName a value relating to the name of the application.  
      *Note: This is necessary to prevent the PSADT entering the ZeroConfigMSI section of the script.*
   2. In the PRE-INSTALLATION section of the script, update the -CloseApps parameter with the executables that should be closed when the installation starts.  
      *Note: This cannot be left blank.*
   3. In the PRE-UNINSTALLATION section of the script, update the -CloseApps parameter with the executables that should be closed when the installation starts.  
      *Note: This cannot be left blank.*
3. Run the installer by using the command Deploy-Application.exe -DeploymentType Install or Deploy-Application.exe -DeploymentType Uninstall.

## Additional requirements

### Multiple .exe files in the Files directory

The ZeroConfigExe extension works by searching the Files directory for the first .exe file it comes across and uses that as the installer. In most cases this works as, generally, installers only consist of one .exe file. In cases where there are more .exe files in the Files directory, the extension will look for setup.exe or install.exe and if it finds either of those, they will be used as the installer file. If neither are found, ZeroConfigExe will not proceed with the install as it will not know which .exe file to use.

In this case, it will be necessary for the administrator to specify which .exe file should be used as the installer. This information should be entered in the variable [string]$installerExecutable defined at the beginning of Deploy-Application.ps1.

For example: [string]$installerExecutable = "softwareinstaller.exe"

### Inno Setup installations

Inno Setup installations can accept an answer file; however, they are not mandatory and if one is not provided the silent install will proceed using the default parameters.

To create an answer file, launch the installer from a command prompt using the ‘/SAVEINF="filename"’ switch (i.e. setup.exe /SAVEINF="C:\Package\install.inf"). Complete the install and then copy the answer file to the Files directory of the PSADT along with the installer. The .inf file can have any name, however there must be only one .inf file in the Files directory.

### InstallShield installations

InstallShield installers require an answer file for both installation and uninstallation in order to be installed silently. These are sometimes provided by the manufacturer, however in many cases you will need to create them yourself.

To create an answer file, launch the installer from a command prompt using the ‘/r’ switch (i.e. setup.exe /r). Run through the installation, changing any of options as you see fit - these will be recorded in the answer file. Once you have completed the install, the answer file will be saved to C:\Windows\setup.iss. Move this file out of the Windows directory and rename it install.iss. Next, create an uninstall answer file by launching the installer again, this time using the ‘/r /uninst’ switches (i.e. setup.exe /r /uninst). Run through the uninstall and then move setup.iss out of the Windows folder and rename it uninstall.iss.

Copy both install.iss and uninstall.iss, along with the installer, to the Files directory for use with the PSADT and ZeroConfigExe. The answer files must be called install.iss and uninstall.iss to work with ZeroConfigExe. If these answer files are not found, ZeroConfigExe will not proceed with the install.

### Office 365 click-to-run installations

Office 365 installations require a configuration .xml file when installing. This file determines which apps are installed and which channel your Office 365 client is in, among other things. Full details about what can be included in this configuration file are provided on [Microsoft Docs](https://docs.microsoft.com/en-us/deployoffice/configuration-options-for-the-office-2016-deployment-tool).

Microsoft provides a tool online that can be used to generate an .xml file with all the settings that are needed. That tool can be found [here](https://config.office.com/deploymentsettings).

The configuration file must be placed in the Files directory along with setup.exe and the Office 365 client data files. The configuration file can have any name, however there must be only one .xml file in the Files directory. If no configuration file is found, ZeroConfigExe will not proceed with the install.

## Logging

The ZeroConfigExe extension logs its progress in the same logging location that is configured for the PSADT, which can be set in AppDeployToolkitConfig.xml. All logs written by the extension log under the component “ZeroConfigExe”.

The following details are logged:

* The application vendor, name and version
* Which installer technology an installer is using
* The full installation or uninstallation command that is used
* If multiple .exe files are found, and the administrator must specify which one to use
* If multiple .msu files are found, and the administrator must specify which one to use
* If the installer technology was not detected
* If an uninstall command was not found in the registry
* If a mandatory configuration or answer file is not found

## Digital Signature

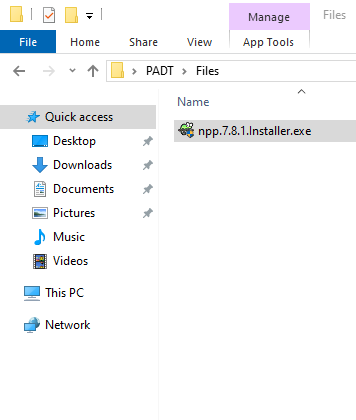
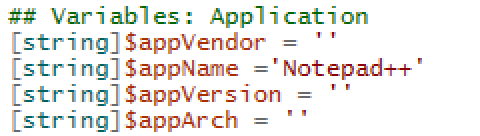
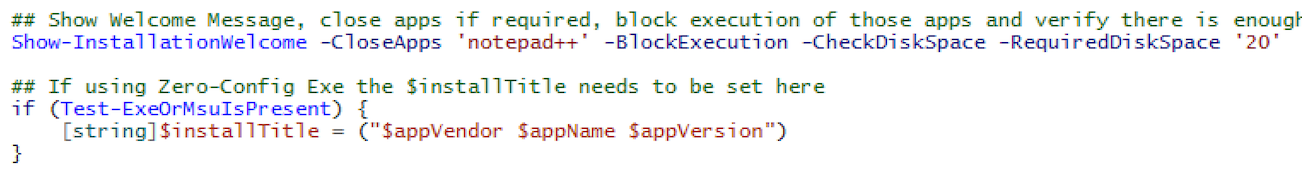
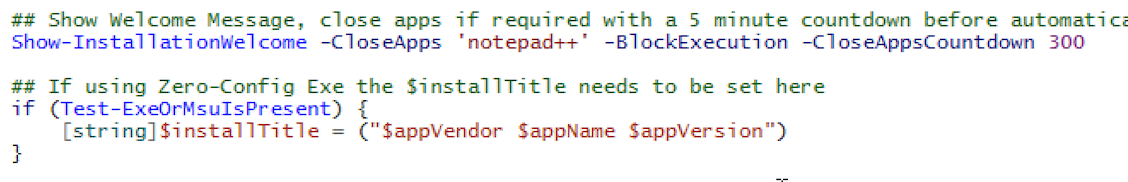
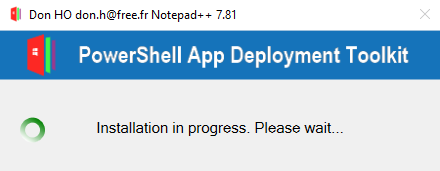
To ensure the copy of ZeroConfigExe you download has not been modified by a malicious third party, the ZeroConfigExeInstallation.ps1 file has been signed by “K Street Consultancy Ltd” (assuming you are downloading from the original project on GitHub and not a fork). If it is not convenient for the script to be signed, or you wish to re-sign it with your own code signing certificate, simply remove the signature block at the end of the script.

Alternatively, you can add the certificate to your Trusted Publishers to allow the script to run on your system(s).

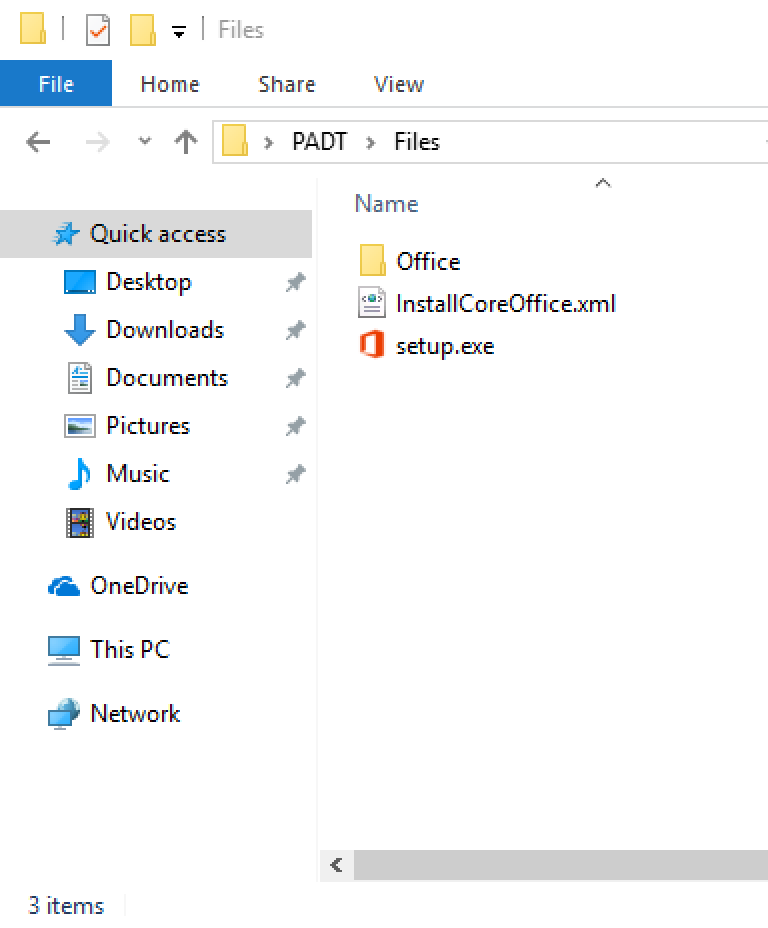
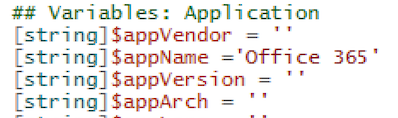
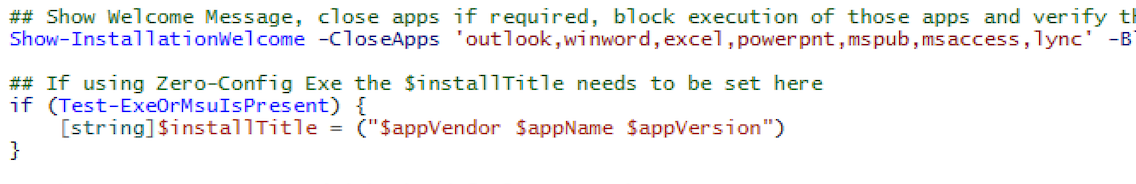
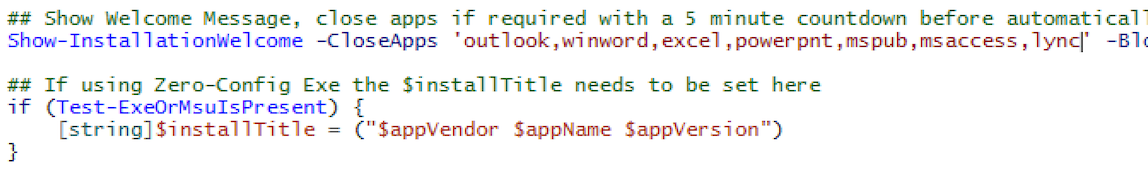
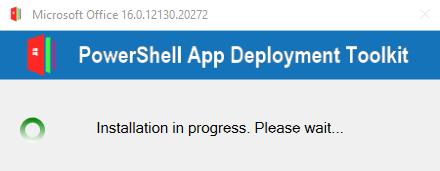
## Example Deployments

### Deploying Notepad++ with ZeroConfigExe

In this example we will silently install Notepad++ using the PowerShell App Deployment Toolkit with the ZeroConfigExe extension.

1. Copy the installer file to the Files directory of the PSADT.  
     
   
2. Open Deploy-Application.ps1 and populate the $appName variable with ‘Notepad++’.  
     
   
3. In the pre-installation section fill out the -CloseApps parameter with ‘notepad++’.  
     
   
4. In the pre-uninstallation section fill out the -CloseApps parameter with ‘notepad++’.  
     
   
5. Install the application by running Deploy-Application.exe.  
     
   

### Deploying Office 365 with ZeroConfigExe

1. Create the configuration .xml file and copy it, along with the Office 365 setup.exe and Office data folder to the File directory of the PSADT.  
     
   
2. Open Deploy-Application.ps1 and populate the $appName variable with ‘Office 365’.  
     
   
3. In the pre-installation section fill out the -CloseApps parameter with ‘outlook,winword,excel,powerpnt,mspub,msaccess,lync’.  
     
   
4. In the pre-uninstallation section fill out the -CloseApps parameter with ‘outlook,winword,excel,powerpnt,mspub,msaccess,lync’.  
     
   
5. Install the application by running Deploy-Application.exe.  
     
   

# Extension Variables

This is a list of the variables that are introduced by the ZeroConfigExe extension which may be used in your own scripts or as part of additional installation / uninstallation logic in Deploy-Application.ps1.

|  |  |
| --- | --- |
| Variable | Description |
| $installerExecutable | Must be set by administrator in Deploy-Application.ps1 if multiple .exe files exist in the Files directory. |
| $overrideDefaultInstall | Can be set to True if the administrator does not wish to use the ZeroConfigExe logic to install an .exe. |
| $defaultExeFile | During an installation, this contains the full path to the installation .exe file found in the Files directory.  During an uninstallation this contains the full path to the uninstall .exe. |
| $defaultMsuFile | Contains the full path to the installation .msu file found in the Files directory. |
| $installerTechnology | Contains the installer technology found for the installer .exe or. msu. |
| $arguments | Contain the arguments used to silently install or uninstall an application. |
| $installISS | When installing an application that uses InstallShield, this contains the full path to the ISS answer file for installing the application. |
| $uninstallISS | When uninstalling an application that uses InstallShield, this contains the full path to the ISS answer file for uninstalling the application. |
| $xmlFile | When installing Office 365 this contains the full path to the .xml configuration file needed for installation. |

# Extension Functions

## Test-ExeOrMsuIsPresent

SYNOPSIS

Check if an .exe or .msu file is present for silent installation/uninstallation.

DESCRIPTION

Check if an .exe or .msu file was found in the same directory as the script or in the Files directory and return true if one was found.

EXAMPLE

Test-ExeOrMsuIsPresent

NOTES

This function is called to automatically determine if this script should be used to silently install or uninstall an application based on the presence or lack of presence of a .exe or .msu file.

## Start-Installation

SYNOPSIS

Invoke the appropriate installer function for the detected installer technology.

DESCRIPTION

Initiates the installer function appropriate for the installer technology detected using the Find-InstallerTechnology function. The detected installer technology is logged to the default log path.

PARAMETER DeploymentType

This parameter is passed through this function so that it can be passed to the appropriate installer function.

EXAMPLE 1

Start-Installation -DeploymentType Install

EXAMPLE 2

Start-Installation -DeploymentType Uninstall

NOTES

This function is the main driver for the script and should be called from the Deploy-Application.ps1 script from the PSADT.

## Find-InstallerTechnology

SYNOPSIS

Find which installer technology a .exe or .msu is using.

DESCRIPTION

Detects which installer technology a .exe is using by searching the .exe for a string of text unique to that installer (i.e. InstallShield for InstallShield installers). For an .msu file the script simply checks for the presence of a .msu file in the script directory or the Files directory.

EXAMPLE

Find-InstallerTechnology

NOTES

Typically, this function is called by the Start-Installation function.

## Find-UninstallStringInRegsitry

SYNOPSIS

Detect the uninstall command for an application in the Windows registry.

DESCRIPTION

Search through the HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall key in the registry to find the uninstall command for an application. On 32bit systems only this key is searched, on 64bit systems both this key and the WOW6432Node key are searched. The results are logged so if no uninstall string is found the user can modify the uninstall behaviour accordingly.

EXAMPLE

Find-UninstallStringInRegsitry

NOTES

This is an internal script function and should typically not be called directly.

## Install-UsingWindowsInstaller

SYNOPSIS

Silently installs an .exe that uses the Windows installer technology.

DESCRIPTION

Sets the arguments needed to silently install a Windows installer application and starts the installation.

For uninstalls this application uses the uninstall command found in the registry.

PARAMETER defaultExeFile

The full path to the executable installer or uninstaller.

PARAMETER deploymentType

Passed from the Start-Installation function and used here to determine if the user wants to install or uninstall the application.

EXAMPLE 1

Install-UsingWixBurnInstaller -defaultExeFile "C:\Product\Setup.exe" -deploymentType Install

EXAMPLE 2

Install-UsingWixBurnInstaller -defaultExeFile "C:\Product\Setup.exe" -deploymentType Uninstall

NOTES

This is an internal script function and should typically not be called directly.

## Install-UsingNSISInstaller

SYNOPSIS

Silently installs an .exe that uses the NSIS installer technology.

DESCRIPTION

Sets the arguments needed to silently install an NSIS application and starts the installation. For uninstalls this application uses the uninstall command found in the registry. NSIS does not output a log file so no log is written.

PARAMETER defaultExeFile

The full path to the executable installer or uninstaller.

PARAMETER DeploymentType

Passed from the Start-Installation function and used here to determine if the user wants to install or uninstall the application.

EXAMPLE 1

Install-UsingNSISInstaller -defaultExeFile "C:\Product\Setup.exe" -DeploymentType Install

EXAMPLE 2

Install-UsingNSISInstaller -defaultExeFile "C:\Product\Setup.exe" -DeploymentType Uninstall

NOTES

This is an internal script function and should typically not be called directly.

## Install-UsingInnoSetupInstaller

SYNOPSIS

Silently installs an .exe that uses the Inno Setup installer technology.

DESCRIPTION

Sets the arguments needed to silently install an NSIS application and starts the installation. Inno Setup supports .inf files with pre-configured settings (such as install path), so if a .inf file is found this is used in the install. For uninstalls this application uses the uninstall command found in the registry. Both install and uninstall log files are written.

PARAMETER defaultExeFile

The full path to the executable installer or uninstaller.

PARAMETER DeploymentType

Passed from the Start-Installation function and used here to determine if the user wants to install or uninstall the application.

EXAMPLE 1

Install-UsingInnoSetupInstaller -defaultExeFile "C:\Product\Setup.exe" -DeploymentType Install

EXAMPLE 2

Install-UsingInnoSetupInstaller -defaultExeFile "C:\Product\Setup.exe" -DeploymentType Uninstall

NOTES

This is an internal script function and should typically not be called directly.

## Install-UsingInstallShieldInstaller

SYNOPSIS

Silently installs an .exe that uses the InstallShield installer technology.

DESCRIPTION

Sets the arguments needed to silently install an InstallShield application and starts the installation. InstallShield requires .iss files for both silent install and uninstall so this is checked and added to the install/uninstall string. The uninstaller uses the same .exe as the installer but with different arguments.

PARAMETER defaultExeFile

The full path to the executable installer or uninstaller.

PARAMETER DeploymentType

Passed from the Start-Installation function and used here to determine if the user wants to install or uninstall the application.

EXAMPLE 1

Install-UsingInstallShieldInstaller -defaultExeFile "C:\Product\Setup.exe" -DeploymentType Install

EXAMPLE 2

Install-UsingInstallShieldInstaller -defaultExeFile "C:\Product\Setup.exe" -DeploymentType Uninstall

NOTES

This is an internal script function and should typically not be called directly.

## Install-UsingWixBurnInstaller

SYNOPSIS

Silently installs an .exe that uses the WiX Burn installer technology.

DESCRIPTION

Sets the arguments needed to silently install a WiX Burn application and starts the installation. The uninstaller uses the same .exe as the installer but with different arguments.

PARAMETER defaultExeFile

The full path to the executable installer or uninstaller.

PARAMETER DeploymentType

Passed from the Start-Installation function and used here to determine if the user wants to install or uninstall the application.

EXAMPLE 1

Install-UsingWixBurnInstaller -defaultExeFile "C:\Product\Setup.exe" -DeploymentType Install

EXAMPLE 2

Install-UsingWixBurnInstaller -defaultExeFile "C:\Product\Setup.exe" -DeploymentType Uninstall

NOTES

This is an internal script function and should typically not be called directly.

## Install-UsingWiseInstaller

SYNOPSIS

Silently installs an .exe that uses the Wise installer technology.

DESCRIPTION

Sets the arguments needed to silently install a Wise application and starts the installation. For uninstalls this application uses the uninstall command found in the registry. Wise does not output a log file, so no log is written.

PARAMETER defaultExeFile

The full path to the executable installer or uninstaller.

PARAMETER DeploymentType

Passed from the Start-Installation function and used here to determine if the user wants to install or uninstall the application.

EXAMPLE 1

Install-UsingWiseInstaller -defaultExeFile "C:\Product\Setup.exe" -DeploymentType Install

EXAMPLE 2

Install-UsingWiseInstaller -defaultExeFile "C:\Product\Setup.exe" -DeploymentType Uninstall

NOTES

This is an internal script function and should typically not be called directly.

## Install-UsingInstallAWARE

SYNOPSIS

Silently installs an .exe that uses the InstallAWARE installer technology.

DESCRIPTION

Sets the arguments needed to silently install an InstallAWARE application and starts the installation.

The uninstaller uses the same .exe as the installer but with different arguments.

PARAMETER defaultExeFile

The full path to the executable installer or uninstaller.

PARAMETER deploymentType

Passed from the Start-Installation function and used here to determine if the user wants to install or uninstall the application.

EXAMPLE 1

Install-UsingInstallAWARE -defaultExeFile "C:\Product\Setup.exe" -deploymentType Install

EXAMPLE 2

Install-UsingInstallAWARE -defaultExeFile "C:\Product\Setup.exe" -deploymentType Uninstall

NOTES

This is an internal script function and should typically not be called directly.

## Install-UsingInstall4j

SYNOPSIS

Silently installs an .exe that uses the Install4j installer technology.

DESCRIPTION

Sets the arguments needed to silently install an Install4j application and starts the installation.

For uninstalls this application uses the uninstall executable found in the registry, which is combined with silent switches and log file location.

PARAMETER defaultExeFile

The full path to the executable installer or uninstaller.

PARAMETER deploymentType

Passed from the Start-Installation function and used here to determine if the user wants to install or uninstall the application.

EXAMPLE 1

Install-UsingInstall4j -defaultExeFile "C:\Product\Setup.exe" -deploymentType Install

EXAMPLE 2

Install-UsingInstall4j -defaultExeFile "C:\Product\Setup.exe" -deploymentType Uninstall

NOTES

This is an internal script function and should typically not be called directly.

## Install-UsingSetupFactory

SYNOPSIS

Silently installs an .exe that uses the Setup Factory installer technology.

DESCRIPTION

Sets the arguments needed to silently install an Install4j application and starts the installation.

For uninstalls this application uses the uninstall executable found in the registry, which is combined with silent switches and log file location.

PARAMETER defaultExeFile

The full path to the executable installer or uninstaller.

PARAMETER deploymentType

Passed from the Start-Installation function and used here to determine if the user wants to install or uninstall the application.

EXAMPLE 1

Install-UsingInstall4j -defaultExeFile "C:\Product\Setup.exe" -deploymentType Install

EXAMPLE 2

Install-UsingInstall4j -defaultExeFile "C:\Product\Setup.exe" -deploymentType Uninstall

NOTES

This is an internal script function and should typically not be called directly.

## Install-UsingOffice365ClickToRunInstaller

SYNOPSIS

Silently installs Microsoft Office 365 click-to-run.

DESCRIPTION

This is one of the special functions designed to install a specific product. It sets the arguments needed to silently install Office 365, which requires the presence of an .xml file which configures the product. For more information see this document from Microsoft: <https://docs.microsoft.com/en-us/deployoffice/deploy-office-365-proplus-from-a-local-source>. For uninstalls this application uses the uninstall command found in the registry.

PARAMETER defaultExeFile

The full path to the executable installer or uninstaller.

PARAMETER DeploymentType

Passed from the Start-Installation function and used here to determine if the user wants to install or uninstall the application.

EXAMPLE 1

Install-UsingOffice365ClickToRunInstaller -defaultExeFile "C:\Office365\Setup.exe" -DeploymentType Install

EXAMPLE 2

Install-UsingOffice365ClickToRunInstaller -defaultExeFile "C:\Office365\Setup.exe" -DeploymentType Uninstall

NOTES

This is an internal script function and should typically not be called directly.

## Install-UsingWindowsUpdateStandaloneInstaller

SYNOPSIS

Silently installs Microsoft Windows update files.

DESCRIPTION

This is one of the special functions designed to install a specific product. It sets the arguments needed to silently install a Windows update. The uninstaller uses the same .msu as the installer but with different arguments. Windows update install logs are written in .evtx format which can be opened in Windows Event Viewer.

PARAMETER defaultExeFile

The full path to the executable installer or uninstaller.

PARAMETER DeploymentType

Passed from the Start-Installation function and used here to determine if the user wants to install or uninstall the application.

EXAMPLE 1

Install-UsingWindowsUpdateStandaloneInstaller -defaultExeFile "wusa.exe" -DeploymentType Install

EXAMPLE 2

Install-UsingWindowsUpdateStandaloneInstaller -defaultExeFile "wusa.exe" -DeploymentType Uninstall

NOTES

This is an internal script function and should typically not be called directly.

## Test-SupportedInstallerType

SYNOPSIS

Allows a user of this script to test whether or not their application will be supported.

DESCRIPTION

This function is used when a user runs this script and specifies the TestSupportedInstallerTypePath variable at the command line, along with a path to an installer executable.

It will output the installer type that is used if it is known by the script, or inform the user that this script does not natively support the installer.

EXAMPLE

ZeroConfigExeInstallation.ps1 -TestSupportedInstallerTypePath "C:\Temp\Application\setup.exe"

NOTES

This is an internal script function and should typically not be called directly. It should be used as shown in the example.