PowerShell App Deployment Toolkit

Administrator Guide

<http://psappdeploytoolkit.codeplex.com>

Contents

[2 Overview 4](#_Toc363645508)

[Introduction 4](#_Toc363645509)

[Features 4](#_Toc363645510)

[Licensing 4](#_Toc363645511)

[3 Toolkit Functionality 5](#_Toc363645512)

[User Interface 5](#_Toc363645513)

[Functions/Logic 5](#_Toc363645514)

[Integration with SCCM 6](#_Toc363645515)

[Help Console 6](#_Toc363645516)

[4 Toolkit Components 6](#_Toc363645517)

[Toolkit File Structure 7](#_Toc363645518)

[Files 7](#_Toc363645519)

[Directories 8](#_Toc363645520)

[Toolkit User Interface 9](#_Toc363645521)

[Balloon tip notifications 9](#_Toc363645522)

[Installation Progress 9](#_Toc363645523)

[Close Application Prompt 10](#_Toc363645524)

[Block Application Execution 11](#_Toc363645525)

[Custom Dialog box 11](#_Toc363645526)

[5 Toolkit Usage 12](#_Toc363645527)

[Overview 12](#_Toc363645528)

[Example Deployment 12](#_Toc363645529)

[Deploying Adobe Reader with the PowerShell App Deployment Toolkit 13](#_Toc363645530)

[Launching the Toolkit 14](#_Toc363645531)

[Overview 14](#_Toc363645532)

[Toolkit Parameters 14](#_Toc363645533)

[Configuring the Toolkit 15](#_Toc363645534)

[6 Toolkit Functions 15](#_Toc363645535)

[Convert-RegistryPath 15](#_Toc363645536)

[Copy-File 16](#_Toc363645537)

[Execute-MSI 16](#_Toc363645538)

[Execute-Process 17](#_Toc363645539)

[Exit-Script 19](#_Toc363645540)

[Get-FileVersion 19](#_Toc363645541)

[Get-HardwarePlatform 20](#_Toc363645542)

[Get-IniContent 20](#_Toc363645543)

[Get-InstalledApplication 20](#_Toc363645544)

[Get-RegistryKey 21](#_Toc363645545)

[Get-ScheduledTask 21](#_Toc363645546)

[Install-MSUpdates 22](#_Toc363645547)

[New-Shortcut 22](#_Toc363645548)

[Refresh-Desktop 23](#_Toc363645549)

[Register-DLL 23](#_Toc363645550)

[Remove-File 23](#_Toc363645551)

[Remove-MSIApplications 24](#_Toc363645552)

[Remove-RegistryKey 24](#_Toc363645553)

[Set-IniContent 25](#_Toc363645554)

[Set-PinnedApplication 25](#_Toc363645555)

[Show-BalloonTip 26](#_Toc363645556)

[Show-DialogBox 26](#_Toc363645557)

[Show-InstallationProgress 27](#_Toc363645558)

[Stop-RunningApplications 28](#_Toc363645559)

[Test-Battery 29](#_Toc363645560)

[Test-MSUpdates 30](#_Toc363645561)

[Test-NetworkConnection 30](#_Toc363645562)

[Test-PowerPoint 30](#_Toc363645563)

[Unregister-DLL 31](#_Toc363645564)

[Update-GroupPolicy 31](#_Toc363645565)

[Write-Log 31](#_Toc363645566)

# Overview

## Introduction

The PowerShell App Deployment Toolkit provides a set of functions to perform common application deployment tasks and to interact with the user during a deployment. It simplifies the complex scripting challenges of deploying applications in the enterprise, provides a consistent deployment experience and improves installation success rates.

## Features

**Easy To Use -** Any PowerShell beginner can use the template and the functions provided with the Toolkit to perform application deployments.

**Consistent -** Provides a consistent look and feel for all application deployments, regardless of complexity.

**Powerful -** Provides a set of functions to perform common deployment tasks, such as installing or uninstalling multiple applications, prompting users to close apps, setting registry keys, copying files, etc.

**User Interface -** Provides user interaction through customizable user interface dialogs boxes, progress dialogs and balloon tip notifications.

**Localized** - The UI is localized in several languages and more can easily be added using the XML configuration file.

**Integration -** Integrates well with SCCM 2007/2012; provides installation and uninstallation deployment types with options on how to handle exit codes, such as supressing reboots or returning a fast retry code.

**Updatable -** The logic engine and functions are separated from per-application scripts, so that you can update the toolkit when a new version is released and maintain backwards compatibility with your deployment scripts.

**Extensible -** The Toolkit can be easily extended to add custom scripts and functions.

**Helpful -** The Toolkit provides detailed logging of all actions performed and even includes a graphical console to browse the help documentation for the Toolkit functions.

## Licensing

The PowerShell App Deployment Toolkit is provided under the Microsoft Public License:

<https://psappdeploytoolkit.codeplex.com/license>

# Toolkit Functionality

## User Interface

* An interface to prompt the user to close specified applications that are open prior to starting the application deployment. The user is prompted to save their documents and has the option to close the programs themselves, have the toolkit close the programs for them or defer the installation (if the AllowDefer option is enabled).
* The ability to prevent the user from launching the applications that need to be closed while the application installation is in progress.
* An indeterminate progress dialog with customizable message text that can be updated throughout the deployment.
* Balloon tip notifications to indicate the beginning and end of an installation and the success or failure of an installation.
* The ability to run in interactive, silent (no dialogs) or non-interactive mode (default for running SCCM task sequence or session 0).
* Branding of the above UI components using a custom logo icon for your own Organization.
* Custom dialog boxes with options to customize title, text, buttons, icon.
* The UI is localized in several languages and more can easily be added using the XML configuration file.

## Functions/Logic

* Provides extensive logging of both the Toolkit functions and any MSI installation / uninstallation.
* Provides the ability to execute any type of setup (MSI or EXEs) and handle the return codes.
* Mass remove MSI applications with a partial match (e.g. remove all versions of all MSI applications which match "Office")
* Perform SCCM actions such as Machine and User Policy Refresh, Inventory Update and Software Update
* Update Group Policy
* Copy / Delete Files
* Get / Set / Remove Registry Keys and Values
* Get / Set Ini File Keys and Values
* Check File versions
* Pin or Unpin applications to the Start Menu or Task Bar
* Create Start Menu Shortcuts
* Register / Unregister dll files
* Refresh desktop icons
* Test network connectivity
* Test power connectivity
* Check whether a PowerPoint slideshow is running

## Integration with SCCM

* Handles SCCM exit codes, including time sensitive dialogs supporting SCCM's Fast Retry feature - providing more accurate SCCM Reporting (no more Failed due to timeout errors).
* Ability to prevent reboot codes (3010) from being passed back to SCCM, which would cause a reboot prompt.
* Supports the CM12 application model by providing an install and uninstall deployment type for every deployment script.
* Bundle multiple application installations to overcome the supported limit of 5 applications in the CM12 application dependency chain.
* Compared to compiled deployment packages, e.g. WiseScript, the Toolkit utilises the SCCM cache correctly and SCCM Distribution Point bandwidth more efficiently by using loose files.

## Help Console

* A graphical console for browsing the help documentation for the toolkit functions.

# Toolkit Components

## Toolkit File Structure

### Files

The toolkit is comprised of the following files:

**Deploy-Application.ps1**

Performs the actual install / uninstall and is the only file that needs to be modified, depending on your level of customisation.

**Deploy-Application.exe**

An optional executable that can be used to launch the Deploy-Application.ps1 script without opening a PowerShell console window. Supports passing command-line parameters to the script.

**AppDeployToolkitMain.ps1**

Contains all of the functions and logic used by the installation script. By Separating the logic from the installation script, we can obfuscate away the complex code and make enhancements independently of the installation scripts that contain per-application actions.

**AppDeployToolkitConfig.xml**

Contains configurable options referenced by the AppDeployToolkit.ps1 script, such as MSI switches and User Interface messages, which are customizable and localized in several languages. This is intended to be a static file that is configured once, not on a per-application basis.

**AppDeployToolkitExtensions.ps1**

This is an optional PowerShell script that can be used to extend the toolkit functionality with custom functions. It is automatically dot-sourced by the AppDeployToolkitMain.ps1 script.

**AppDeployToolkitHelp.ps1**

This is a script that displays a help console to browse the functions included in the Toolkit and copy and paste examples in to your deployment script.



### Directories

The Root folder contains the Deploy-Application.exe and Deploy-Application.ps1 files. The Deploy-Application.ps1 file is the only file that should be modified on a per-application basis.

The directories below contain the installation files and supporting files referenced by the toolkit.

**AppDeployToolkit**

Folder containing the toolkit dependency files.

**Files**

Folder containing your main setup files, e.g. MSI

**Support Files**

Folder containing any supporting files such as files you need to copy to the target machine using the toolkit during deployment.

## Toolkit User Interface

The user interface consists of several components detailed below. All of the UI components include message text that is customizable in the AppDeployToolkitConfig.xml. The UI has been localised in 5 different languages: English, French, Spanish, Portuguese and German. Additional languages can easily be added in the XML configuration file.

The language used by the Toolkit UI is selected automatically based on the language culture of the operating system, so the same AppDeployToolkitConfig file can be used in a multi-language environment.

The user interface can be suppressed by specifying the deploy mode parameter as follows:

**Depl0yApplication.ps1 –DeployMode “Silent”**

### Balloon tip notifications

Balloon tip notifications are displayed in the system tray automtically at the beginning and end of the installation.

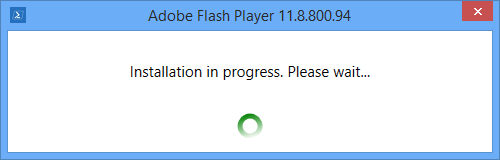




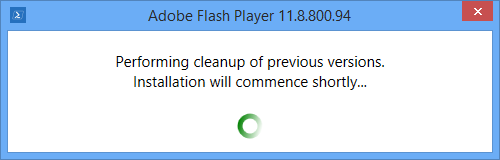


### Installation Progress

The installation progress message displays an indeterminate progress ring to indicate an installation is in progress and display status messages to the end user. This is invoked using the “Show-InstallationProgress” function.

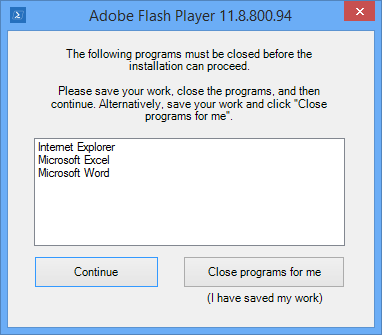


The progress message can be dynamically updated to indicate the stage of the installation or to display custom messages to the user, using the “Show-InstallationProgress” function.



### Close Application Prompt

A close application prompt can be displayed to the user to close running processes before the installation proceeds, using the “Stop-RunningApplications” function.

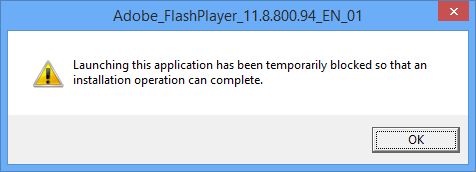


Close prompt with optional defer button:



### Block Application Execution

If the block execution option is enabled (see Stop-RunningApplications function in this document), the user will be prompted that they cannot launch the specified application(s) while the installation is in progress. The application will be unblocked again once the installation has completed.



### Custom Dialog box

A generic dialog box to display custom messages to the user. This can be customized with different icons and buttons and could be used for example to allow the user to defer the entire installation even when a close program prompt is not displayed.





# Toolkit Usage

## Overview

The Deploy-Application.ps1 script is the only script you need to modify to deploy your application.

The Deploy-Application.ps1 is broken down into the following sections:

**Initialization** e.g. Variables such as App Vendor, App Name, App Version

**Pre-Install** e.g. Close applications, uninstall or clean-up previous versions

**Install** e.g. Install the primary application, or components of the application

**Post-Install** e.g. Drop additional files, registry tweaks

**Uninstall** e.g. Uninstall/rollback the changes performed in the install section.

## Example Deployment

### Deploying Adobe Reader with the PowerShell App Deployment Toolkit

1. Copy the application source files in to the “Files” directory, e.g.



1. Customize the Deploy-Application.ps1 script using the example code below
2. Install the application by running Deploy-Application.ps1
3. Uninstall the application by running Deploy-Application.ps1 –DeploymentType “Uninstall”

**Initialization**

# Populate these variables with the application and script details:

$appVendor = "Adobe"

$appName = "Reader"

$appVersion = "11.0.3"

$appArch = ""

$appLang = "EN"

$appRevision = "01"

$appScriptVersion = "1.0.0"

$appScriptDate = "08/07/2013"

$appScriptAuthor = "Your Name”

**Pre-Install**

# Prompt the user to close the following applications if they are running:

Stop-RunningApplications "iexplore,AcroRd32,cidaemon"

# Show Progress Message (with the default message)

Show-InstallationProgress

# Remove any previous versions of Adobe Reader

Remove-MSIApplications "Adobe Reader"

**Installation**

# Install the base MSI and apply a transform

Execute-MSI -Action Install -Path "Adobe\_Reader\_11.0.0\_EN.msi" -Transform "Adobe\_Reader\_11.0.0\_EN\_01.mst"

# Install the patch

Execute-MSI -Action Patch -Path "Adobe\_Reader\_11.0.3\_EN.msp"

**Post-Installation**

# No actions required here

**Uninstallation**

# Prompt the user to close the following applications if they are running:

Stop-RunningApplications "iexplore,AcroRd32,cidaemon"

# Show Progress Message (with a message to indicate the application is being uninstalled)

Show-InstallationProgress -StatusMessage "Uninstalling Application $installTitle. Please Wait..."

# Remove this version of Adobe Reader

Execute-MSI -Action Uninstall -Path "{AC76BA86-7AD7-1033-7B44-AB0000000001}"

## Launching the Toolkit

### Overview

There are two ways to launch the toolkit for deployment of applications.

1. Launch “Deploy-Application.ps1” PowerShell script as administrator.
2. Launch “Deploy-Application.exe” as administrator. This will launch the “Deploy-Application.ps1” PowerShell script without opening a PowerShell command window. Note, if the x86 PowerShell is required (for example, if CAPICOM or another x86 library is needed), launch **Deploy-Application.exe /32**

#### Examples:

**Deploy-Application.ps1**

*Deploy an application for installation*

**Deploy-Application.ps1 –DeploymentType “Uninstall” –DeployMode “Silent”**

*Deploy an application for uninstallation in silent mode*

**Deploy-Application.exe –AllowRebootPassThru**

*Deploy an application for installation, supressing the PowerShell console window and allowing reboot codes to be returned to the parent process.*

**Deploy-Application.exe /32 –DeploymentType “Uninstall” –DeployMode “Silent”**

*Deploy an application for uninstallation using PowerShell x86, supressing the PowerShell console window and deploying in silent mode.*

### Toolkit Parameters

The following parameters are accepted by Deploy-Application.ps1:

**-DeploymentType** “Install” | “Uninstall” (default is install)

Specifies whether to install or uninstall the application.

**-DeployMode** “Interactive” | “Silent” | “NonInteractive” (default is interactive)

Specifies whether the installation should be run in Interactive, Silent or NonInteractive mode.

Silent = No dialogs (progress and balloon tip notifications are supressed)

NonInteractive = Very silent, i.e. no blocking apps. Noninteractive mode is automatically set if an SCCM task sequence or session 0 is detected.

**-AllowRebootPassThru** $true | $false (default is false)

Specifies whether to allow the 3010 exit code (reboot required) to be passed back to the parent process (e.g. SCCM) if detected during an installation. If a 3010 code is passed to SCCM, the SCCM client will display a reboot prompt. If set to false, the 3010 return code will be replaced by a “0” (successful, no restart required).

## Configuring the Toolkit

Aside from customizing the “Deploy-Application.ps1” script to deploy your application, no configuration is necessary out of the box. The following components can be configured as required:

**AppDeployToolkitConfig.xml** - Configure the default UI messages, MSI parameters and log file location.

**AppDeployToolkitLogo.ico** - To brand the toolkit with your own custom/corporate logo, replace the AppDeployToolkitLogo.ico file with your own .ico file (retaining the file name)

# Toolkit Functions

### Convert-RegistryPath

**Synopsis :** Converts the specified registry key path to a format that is compatible with built-in PowerShell cmdlets.

**Description :**

Converts the specified registry key path to a format that is compatible with built-in PowerShell cmdlets.

Converts registry key hives to their full paths, e.g. HKLM is converted to "HKEY\_LOCAL\_MACHINE" and prepends "Registry::" to the path

**Parameter :** Key

Path to the registry key to convert (can be a registry hive or fully qualified path)

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Convert-RegistryPath -Key "HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{1AD147D0-BE0E-3D6C-AC11-64F6DC4163F1}"

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Convert-RegistryPath -Key "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{1AD147D0-BE0E-3D6C-AC11-64F6DC4163F1}"

### Copy-File

**Synopsis :** Function to copy a file to a destination path.

**Description :**

Function to copy a file to a destination path.

**Parameter :** Path

Path of the file you want to copy

Destination

Destination Path of the file to copy

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Copy-File -Path "$dirSupportFiles\MyApp.ini" -Destination "$envWindir\MyApp.ini"

### Execute-MSI

**Synopsis :** Executes msiexec.exe to perform the following actions for MSI & MSP files and MSI product codes: install, uninstall, patch, repair, active setup.

**Description :**

Executes msiexec.exe to perform the following actions for MSI & MSP files and MSI product codes: install, uninstall, patch, repair, active setup.

Sets default switches to be passed to msiexec based on the preferences in the XML configuration file, e.g. "REBOOT=ReallySuppress /QB!"

Automatically generates a log file name and creates a verbose log file for all msiexec operations.

NB: Expects the MSI or MSP file to be located in the "Files" sub directory of the App Deploy Toolkit. Expects transform files to be in the same directory as the MSI file.

**Parameter :** Action

The action to perform ["Install","Uninstall","Patch","Repair","ActiveSetup"]

Path

The path to the MSI/MSP file or the product code of the installed MSI.

Transform

The name of the transform file(s). The transform file is expected to be in the same directory as the MSI file.

Parameters

Overrides the default parameters specified in the XML configuration file. Install default is "REBOOT=ReallySuppress /QB!", uninstall default is "REBOOT=ReallySuppress /QN"

LogName

Overrides the default log file name.

The default log file name is generated from the MSI file name or for uninstallations, the product code is resolved to the displayname and version of the application.

WorkingDirectory

Overrides the working directory.

The working directory is set to the location of the MSI file.

ContinueOnError

Continue if an exit code is returned by msiexec that is not recognised by the App Deploy Toolkit.

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Give an example of how to use it

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Give another example of how to use it

### Execute-Process

**Synopsis :** Function to execute a process, with optional arguments, working directory, window style.

**Description :**

Executes a process, e.g. a file included in the Files directory of the App Deploy Toolkit, or a file on the local machine.

Provides various options for handling the return codes (see Parameters)

**Parameter :** FilePath

Path of the file you want to execute.

If the file is located directly in the "Files" directory of the App Deploy Toolkit, only the file name needs to be specified.

Otherwise, the full path of the file must be specified. If the files is in a subdirectory of "Files", use the "$dirFiles" variable as shown in the example above.

Arguments

Arguments to be passed to the executable

WindowStyle

Style of the window of the process executed: "Normal","Hidden","Maximized","Minimized" [Default is "Normal"]

WorkingDirectory

The working directory used for executing the process.

Defaults to the directory of the file being executed.

PassThru

Returns STDOut and STDErr output from the process.

IgnoreExitCodes

List the exit codes you want to ignore.

ContinueOnError

Continue if an exit code is returned by the process that is not recognised by the App Deploy Toolkit.

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Execute-Process -FilePath "uninstall\_flash\_player\_64bit.exe" -Arguments "/uninstall" -WindowStyle Hidden

If the file is in the "Files" directory of the App Deploy Toolkit, only the file name needs to be specified.

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Execute-Process -FilePath "$dirFiles\Bin\setup.exe" -Arguments "/S" -WindowStyle Hidden

-------------------------- EXAMPLE 3 --------------------------

C:\PS>Execute-Process -FilePath "setup.exe" -Arguments "/S" -IgnoreExitCodes "1,2"

### Exit-Script

**Synopsis :** This function exits the scripts, performs cleanup actions and passes an exit code to the parent process.

**Description :**

This function should always be used when exiting the script, to ensure cleanup actions are performed.

This function performs cleanup actions, such as closing down dialogs and unblocking blocked applications.

It displays a balloon tip notification to indicate the setup is complete and whether it was a success or a failure.

The function determines what exit code to pass to the parent process depending on the the options specified in the deployment script, e.g.

If $AllowRebootPassThru is set to False, it will suppress any "3010" exit codes detected during the installation and instead pass the "0" exit code.

**Parameter :** ExitCode

The exit code to be passed from the script to the parent process, e.g. SCCM

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Exit-Script -ExitCode "0"

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Exit-Script -ExitCode "1618"

### Get-FileVersion

**Synopsis :** Gets the version of the specified file

**Description :**

Gets the version of the specified file

**Parameter :** File

Path of the file

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Get-FileVersion "$envProgramFilesX86\Adobe\Reader 11.0\Reader\AcroRd32.exe"

### Get-HardwarePlatform

**Synopsis :** Retrieves information about the hardware platform (physical or virtual)

**Description :**

Retrieves information about the hardware platform (physical or virtual)

**Parameter :**

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Get-HardwarePlatform

### Get-IniContent

**Synopsis :** Parses an ini file and returns the contents as objects with ini section, name and value properties

**Description :**

Parses an ini file and returns the contents as objects with ini section, name and value properties

**Parameter :** FilePath

Path to the ini file

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Get-IniContent "$envProgramFilesX86\IBM\Lotus\Notes\notes.ini"

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Get-IniContent "$envProgramFilesX86\IBM\Lotus\Notes\notes.ini" | Where { $\_.Name -eq "KeyFileName" } | Select Value -ExpandProperty Value

### Get-InstalledApplication

**Synopsis :** Retrieves information about installed applications.

**Description :**

Retrieves information about installed applications by querying the registry. You can specify an application name, a product code, or both.

Returns information about application publisher, name & version, product code, uninstall string, install source, location & date.

**Parameter :** Name

ProductCode

The product code of the application you want to retrieve information on.

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Get-InstalledApplication -Name "Adobe Flash"

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Get-InstalledApplication -ProductCode "{1AD147D0-BE0E-3D6C-AC11-64F6DC4163F1}"

### Get-RegistryKey

**Synopsis :** Retrieves value names and value data for a specified registry key

**Description :**

Retrieves value names and value data for a specified registry key.

If the registry key does not contain any values, the function will return $null. If you need to test for existence of a registry key path, use the built-in Test-Path cmdlet

**Parameter :** Key

Path of the registry key

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Get-RegistryKey "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{1AD147D0-BE0E-3D6C-AC11-64F6DC4163F1}"

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Get-RegistryKey "HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\iexplore.exe"

### Get-ScheduledTask

**Synopsis :** Retrieves a list of the scheduled tasks on the local computer

**Description :**

Retrieves a list of the scheduled tasks on the local computer and returns them as an array

**Parameter :**

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Get-ScheduledTask

### Install-MSUpdates

**Synopsis :** Installs all Microsft Updates in a given directory

**Description :**

Installs all Microsft Updates in a given directory of type ".exe", ".msu" or ".msp"

**Parameter :** Directory

Directory containing the updates

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Install-MSUpdate "$dirFiles\MSUpdates"

### New-Shortcut

**Synopsis :** Creates a new shortcut .lnk file, which can be used for example on the start menu.

**Description :**

Creates a new shortcut .lnk file, with configurable options.

**Parameter :** Path

Path to save the shortcut

TargetPath

Target path that the shortcut launches

Arguments

Arguments to be passed to the target path

IconLocation

Location of the icon used for the shortcut

Description

Description of the shortcut

WorkingDirectory

Working Directory to be used for the target path

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>New-Shortcut -Path "$envProgramData\Microsoft\Windows\Start Menu" -TargetPath "$envWinDir\system32\notepad.exe" -IconLocation "$envWinDir\system32\notepad.exe" -Description "Notepad" -WorkingDirectory

"$envHomeDrive\$envHomePath"

### Refresh-Desktop

**Synopsis :** Forces the Windows Exporer Shell to refresh, which causes desktop icons to be reloaded

**Description :**

Forces the Windows Exporer Shell to refresh, which causes desktop icons to be reloaded.

Informs the Explorer Shell to refresh its settings after you change registry values or other settings to avoid a reboot.

**Parameter :**

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Refresh-Desktop

### Register-DLL

**Synopsis :** Registers a DLL file

**Description :**

Registers a DLL file using regsvr32.exe

**Parameter :** FilePath

Path to the DLL file

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Register-DLL "$envProgramFiles\Documentum\Shared\DcTLSFileToDMSComp.dll"

### Remove-File

**Synopsis :** Function to remove a file or all files recursively in a given path.

**Description :**

Function to remove a file or all files recursively in a given path.

**Parameter :** Path

Path of the file you want to remove

Recurse

Optionally, remove all files recursively in a directory

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Remove-File -Path "C:\Windows\Downloaded Program Files\Temp.inf"

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Remove-File -Path "C:\Windows\Downloaded Program Files" -Recurse

### Remove-MSIApplications

**Synopsis :** Removes all MSI applications matching the specified application name

**Description :**

Removes all MSI applications matching the specified application name.

Enumerates the registry for installed applications matching the specified application name and uninstalls that application using the product code, provided the uninstall string matches "msiexec"

**Parameter :** Name

The name of the application you want to uninstall.

ContinueOnError

Continue if an exit code is returned by msiexec that is not recognised by the App Deploy Toolkit.

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Remove-MSIApplications "Adobe Flash"

Removes all versions of software that match the name "Adobe Flash"

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Remove-MSIApplications "Adobe"

Removes all versions of software that match the name "Adobe"

### Remove-RegistryKey

**Synopsis :** Deletes the specified registry key or value

**Description :**

Deletes the specified registry key or value

**Parameter :** Key

Path of the registry key to delete

Name

Name of the registry key value to delete

Recurse

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Remove-RegistryKey -Key "HKEY\_CURRENT\_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Remove-RegistryKey -Key "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" -Name "RunAppInstall"

### Set-IniContent

**Synopsis :** Adds or sets the value of a property in an ini file

**Description :**

Adds or sets the value of a property in an ini file

**Parameter :** FilePath

Path to the inin file

Key

The ini property name

Value

The ini property value

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Set-IniContent "$envProgramFilesX86\IBM\Lotus\Notes\notes.ini" -Key "AutoLogoffMinutes" -Value "10"

### Set-PinnedApplication

**Synopsis :** Pins or unpins a shortcut to the start menu or task bar.

**Description :**

Pins or unpins a shortcut to the start menu or task bar.

This should typically be run in the user context, as pinned items are stored in the user profile.

**Parameter :** Action

Action to be performed: "PintoStartMenu","UnpinfromStartMenu","PintoTaskbar","UnpinfromTaskbar"

FilePath

Path to the shortcut file to be pinned or unpinned

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Set-PinnedApplication -Action "PintoStartMenu" -FilePath "$envProgramFilesX86\IBM\Lotus\Notes\notes.exe"

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Set-PinnedApplication -Action "UnpinfromTaskbar" -FilePath "$envProgramFilesX86\IBM\Lotus\Notes\notes.exe"

### Show-BalloonTip

**Synopsis :** Displays a balloon tip notification in the system tray

**Description :**

Displays a balloon tip notification in the system tray

**Parameter :** BalloonTipText

Text of the balloon tip

BalloonTipTitle

Title of the balloon tip

BalloonTipIcon

Icon to be used [Default is Info]

Accepted values: 'Error', 'Info', 'None', 'Warning'

BalloonTipTime

Time in milliseconds to display the balloon tip [Default 500]

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Show-BalloonTip -BalloonTipText "Installation Started" -BalloonTipTitle "Application Name"

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Show-BalloonTip -BalloonTipIcon "Info" -BalloonTipText "Installation Started" -BalloonTipTitle "Application Name" -BalloonTipTime "1000"

### Show-DialogBox

**Synopsis :** This function displays a custom dialog box with optional title, buttons, icon and timeout.

**Description :**

This function displays a custom dialog box with optional title, buttons, icon and timeout. The default button is "OK", the default Icon is "None" and the default Timeout is none.

**Parameter :** Text

Text in the message dialog box

Title

Title of the message dialog box

Buttons

Buttons to be included on the dialog box [Default is "OK"]

"OK"

"OKCancel"

"AbortRetryIgnore"

"YesNoCancel"

"YesNo"

"RetryCancel"

"CancelTryAgainContinue"

Icon

Icon to display on the dialog box [Default is "None"]

Acceptable valures are: "None", "Stop", "Question", "Exclamation", "Information",

Timeout

Timeout period in seconds before automatically closing the dialog box with the return message "Timeout" [Default is None]

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Show-DialogBox -Title "Installed Complete" -Text "Installation has completed. Please click OK and restart your computer." -Icon "Information"

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Show-DialogBox -Title "Installation Notice" -Text "Installation will take approximately 30 mintues. Do you wish to proceed?" -Buttons "OKCancel" -Icon "Exclamation" -Timeout 600

### Show-InstallationProgress

**Synopsis :** Displays a progress dialog in a separate thread with an updatable custom message.

**Description :**

Create a WPF window in a separate thread to display a marquee style progress ellipse with a custom message that can be updated.

The status message supports line breaks.

The first time this function is called in a script, it will display a balloon tip notification to indicate that the installation has started.

**Parameter :** statusMessage

The Status Message to be displayed. The default status message is taken from the XML configuration file.

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Show-InstallationProgress

Uses the default status message from the XML configuration file.

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Show-InstallationProgress "Installation in Progress..."

-------------------------- EXAMPLE 3 --------------------------

C:\PS>Show-InstallationProgress "Installation in Progress...`nThe installation may take 20 minutes to complete."

### Stop-RunningApplications

**Synopsis :** This function prompts the user to close the specified running applications or optionally closes the applications without showing a prompt (using the -Silent" switch).

**Description :**

The user is presented with a dialog box to close the applications themselves and continue or to have the script close the applications for them.

If the -AllowDefer option is specified, an optional "Defer" button will be shown to the user. If the user selects this option, the script will exit and return a 1618 code (SCCM fast retry code)

Optionally, by using the -Silent switch, you can stop running processes without prompting the user at all.

By specifying the -BlockExecution option, the user will be prevented from launching those same applications while the installation is in progress.

The process descriptions are retrieved from WMI, with a fall back on the process name if no description is available. Alternatively, you can specify the description yourself with a '=' symbol - see examples.

The dialog box will timeout after 1 hour and 55 minutes to prevent SCCM installations from timing out and returning a failure code to SCCM. When the dialog times out, the script will exit and return a 1618

code (SCCM

fast retry code).

**Parameter :** ProcessName

Name of the process to stop (do not include the .exe)

BlockExecution

Option to prevent the user from launching the process/application

AllowDefer

Enables an optional defer button to allow the user to defer the installation if they do not want to close running applications.

Silent

Stop processes without prompting the user

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Stop-RunningApplications "iexplore,winword,excel"

Prompt the user to close Internet Explorer, Word and Excel.

-------------------------- EXAMPLE 2 --------------------------

C:\PS>Stop-RunningApplications "winword,excel" -Silent

Close Word and Excel without prompting the user.

-------------------------- EXAMPLE 3 --------------------------

C:\PS>Stop-RunningApplications "winword,excel" -BlockExecution

Close Word and Excel and prevent the user from launching the applications while the installation is in progress.

-------------------------- EXAMPLE 4 --------------------------

C:\PS>Stop-RunningApplications "winword=Microsoft Office Word,excel=Microsoft Office Excel"

Prompt the user to close Word and Excel, with customized descriptions for the applications.

### Test-Battery

**Synopsis :** Tests whether the local machine is running on battery

**Description :**

Tests whether the local machine is running on battery and returns true/false

**Parameter :**

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Test-Battery

-------------------------- EXAMPLE 2 --------------------------

C:\PS>

### Test-MSUpdates

**Synopsis :** Test whether an Microsoft Windows update is installed

**Description :**

Test whether an Microsoft Windows update is installed

**Parameter :** KBNumber

KBNumber

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Test-MSUpdate "KB2549864"

### Test-NetworkConnection

**Synopsis :** Tests for an active network connection

**Description :**

Tests for an active network connection by querying the Win32\_NetworkAdapter WMI class.

**Parameter :**

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Test-NetworkConnection

### Test-PowerPoint

**Synopsis :** Tests whether Power point is running in presentation mode

**Description :**

Tests whether Power point is running in presentation mode

**Parameter :**

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Test-PowerPoint

### Unregister-DLL

**Synopsis :** Unregisters a DLL file

**Description :**

Unregisters a DLL file using regsvr32.exe

**Parameter :** FilePath

Path to the DLL file

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Unregister-DLL "$envProgramFiles\Documentum\Shared\DcTLSFileToDMSComp.dll"

### Update-GroupPolicy

**Synopsis :** Performs a gpupdate command to refresh Group Policies on the local machine

**Description :**

Performs a gpupdate command to refresh Group Policies on the local machine

**Parameter :**

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Update-GroupPolicy

### Write-Log

**Synopsis :** Writes output to the console and log file simultaneously

**Description :**

This functions outputs text to the console and to the log file specified in the XML configuration.

The date, time and installation phase is pre-pended to the text, e.g. [30-07-2013 11:27:07] [Initialization] "Deploy Application script version is [2.0.0]"

**Parameter :** Text

The text to display in the console and to write to the log file

**Examples :**

-------------------------- EXAMPLE 1 --------------------------

C:\PS>Write-Log -Text "This is a custom message..."