# MECM AddApp Tool

Version: 2.11.1

Written by: Joel Chettle

As of: 9/6/2022

Contents

[MECM AddApp Tool 1](#_Toc113368773)

[Description 1](#_Toc113368774)

[Requirements 1](#_Toc113368775)

[How-to Launch 2](#_Toc113368776)

[Configuration 3](#_Toc113368777)

[How-to Use 5](#_Toc113368778)

[Example 8](#_Toc113368779)

[Troubleshooting 10](#_Toc113368780)

## Description

If consistent packaging standards and naming conventions are in place, this tool allows the packager to automatically create the Active Directory group (if AD is used for software distribution), Collection, Application, and Deployment in SCCM for a package by providing minimal input.

## Requirements

* Microsoft .Net Framework 2.0+
* Microsoft PowerShell 3.0 (minimum); 4.0+ (recommended)
* Active Directory and SCCM modules, which are included automatically if both AdminTools and Configuration Manager are installed. Note: The SCCM cmdlet library may get updated by Microsoft, in which commands within the script may become depreciated or could no longer function.

**Files Required:**

* MECMAddAppTool.ps1
* MECMAddAppSettings.xml (will be generated by the tool)

**Files Included but Optional:**

* Changelog.txt
* LICENSE.txt
* MECMAddAppTool.ico

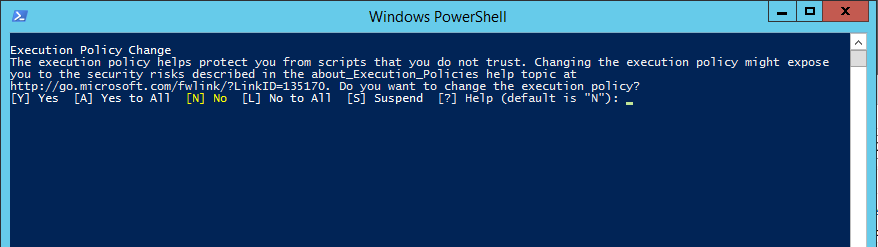
## How-to Launch

To run the tool, simply right-click the ps1 file and select “Run with PowerShell”.

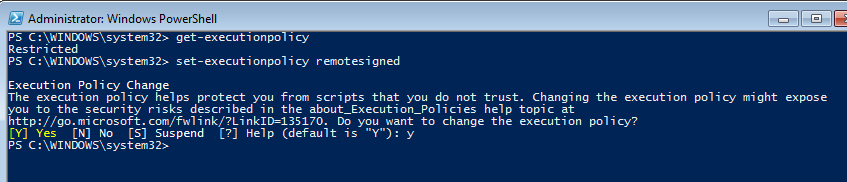
The tool must be run with an admin account that has access to make modifications to the relevant areas of both Active Directory and MECM. To run with a different account from the logged-in user you would have to run PowerShell.exe with a “Run As” and then navigate to the ps1.

The tool used to be provided as a compiled EXE (with the use of PS2EXE-GUI by Markus Scholtes (a rework of PS2EXE by Igor Karstein)). Unfortunately, this regularly would get flagged by anti-virus software as a false positive. So, I have decided by default to leave as a ps1, in which case you will have the option to compile it yourself if you still have the ability and desire. The icon file is included for reference.

Depending on the Execution Policy set on the device, you may get prompted to continue. Enter “Y” if so.



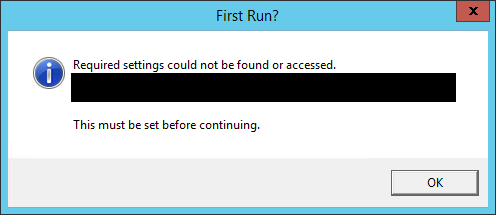
Note: If Execution Policy is set to Restricted, then custom scripts cannot execute. You can verify this by running PowerShell.exe and enter “get-executionpolicy”. You may also change your Execution Policy using “set-executionpolicy”. (You will not have to do this every time. Setting the policy would be permanent.) PowerShell must be run as an admin. I recommend researching this for more information. This policy might also be managed by system administrators.

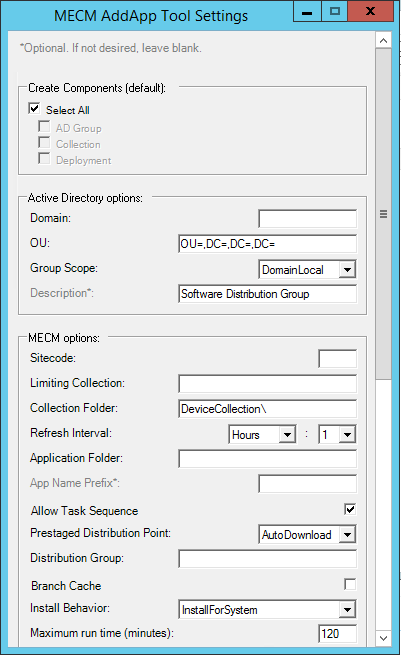
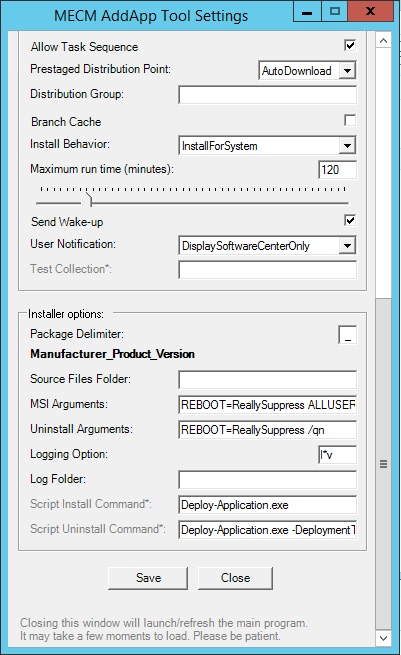


After launching the ps1, the PowerShell console will then automatically close. It may take a moment before the GUI loads. This is normal.

## Configuration

All custom settings are stored in a settings XML. This is so the core (ps1) script does not have to be modified by the user. The settings XML can be created and modified using the tool interface. If the file does not exist, it will automatically prompt you to create one. This will be stored in the same directory as the ps1.



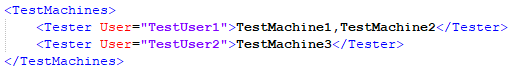
Hovering the cursor over each field should reveal a Tooltip displaying more information.

Enter in data or make selections for each required field, then hit “Save”. After it has confirmed “Saved!” you can hit “Close”, and the main form will open.

After these are set, they will remain consistent for all packages created for the site.

You can review and modify these settings any time from the main form by using File > Settings.

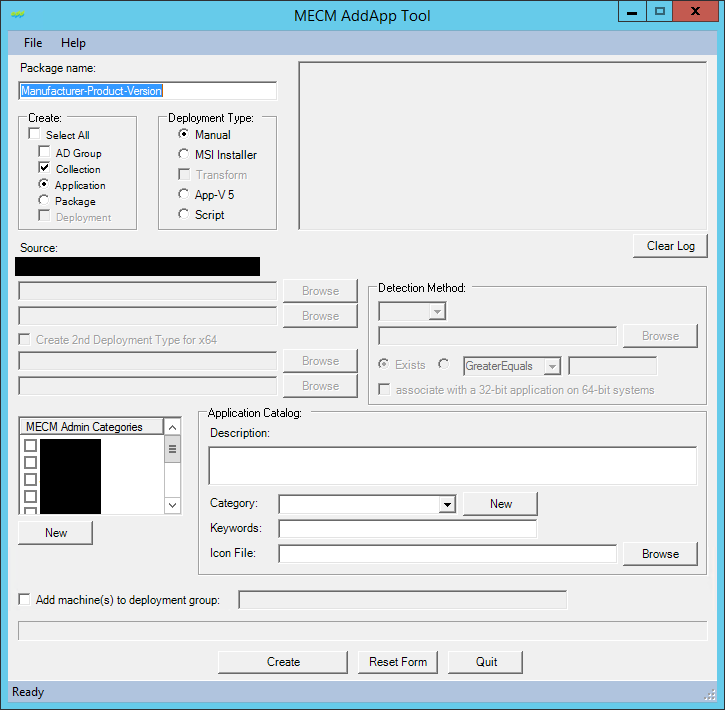
There exists an additional configuration setting that can be entered in the XML that currently does not exist in the form interface. This is to predefine test machines that would be added to the new AD group. It should be entered as following; defining the machine(s) per user, as the tool will associate it to the user ID running the tool. This is to allow different machines for different testers.



Note: Saving settings from the tool interface will overwrite the XML. Any custom entries such as these will be lost.

## How-to Use

Interface:

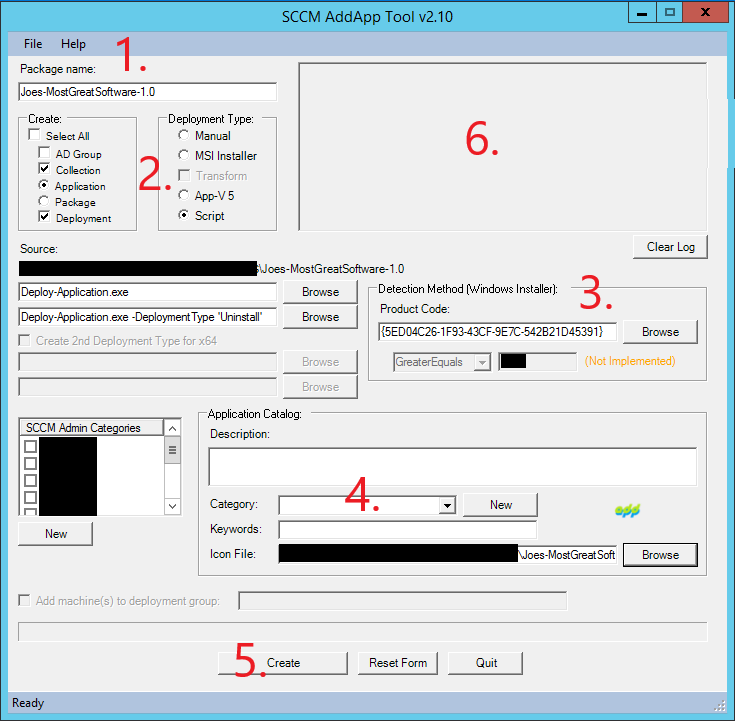


Breakdown:

|  |  |
| --- | --- |
|  | Enter the package name in the textbox. Must be in the format shown. The delimiter (In this case “-”) may be different depending on the entry in the settings XML. The max length for this text field is 50 when selecting a Package or 64 when selecting an Application. |
|  | Define which options you would like the tool to create or Select All to create all. When selecting between Application or Package type, note that the tool was built for Applications. But some functionality is there for Packages. (It will create an empty Package with no Programs.) |
|  | Choose a Deployment type option.   * Manual. This is for situations where the tool cannot create the Deployment type by the other means. You will “manually” create it after the tool sets up everything else. * MSI. This is for packages that are simple MSI’s only, with/without a Transform. Install/Uninstall commands, etc. are built automatically. * App-V 5. This is for App-V5 packages. * Script. This creates a script-based installer in which you can define the install/uninstall commands. |
|  | The Source folder will be pulled from the Settings XML + the Package name. The source files must be under this location.  When selecting MSI type, the top box will be for the MSI. The 2nd box for the MST.  When selecting AppV5, the top box will be for the AppV.  When selecting Script, the top box will be for the Install command. The 2nd box will be for the Uninstall command. This will prepopulate with settings from the XML file.  Browse allows you to navigate to a file. |
|  | If the checkbox for a 2nd Deployment Type is selected for x64, this will allow selection of two sets of MSI/MST’s. Note that the path is changed slightly to include a “x86” and “x64” folder. But Browse is available to change it.  NOTE: This will create two MSI Deployment Types in SCCM. But it will not set the Operating System Requirements. That will have to be a manual setting afterward. |
|  | Detection method will be enabled if selecting a Script type. A dropdown allows the selection of either MSI, File, or Registry detection.  Browse allows you to navigate to a MSI and import the product code or browse to a File path. Browse is not available for Registry at this time.  Can select whether to detect if it Exists, or by Version greater/less/equal/etc. For Registry detection, the Exist option will compare the property as a String. There is also a checkbox for “associate with a 32-bit application on 64-bit systems” for File and Registry checks, which correlates with the like setting in MECM. |
|  | This displays current Admin categories pulled from SCCM. Any or multiple can be selected for the App.  New button dynamically creates a new category in SCCM and will display it at the bottom of the list. |
|  | |
| Fields used for Application Catalog settings.  Description and Keywords are text fields.  Category is a dropdown list. It also pulls current categories defined in SCCM, and like the admin category, allows you to create New. Only one can be selected though.  Icon file is the file to use for the application icon. This can be a .ico or .exe. It cannot use .dll’s. Browse allows you to navigate to the file. Once selected, you will see the image of the icon appear on screen. When selecting an exe, it extracts the image into a png format to %temp%. After the application is created, this file is removed during cleanup. | |
|  | |
| If you wish to add machine names to the newly created AD group, check the box and enter them in the text field.  If Testers are defined in the settings XML, this will prepopulate with those entries.  This is disabled unless AD Group is checked. | |
|  | Create button starts the process after all selections have been made. Once complete, the form will reset.  Reset Form resets all the fields and selections to the form’s original state.  Quit closes the form. |
|  | This box is read-only and will display messages as progress completes. Certain actions will perform validation checks in which will display Pass or Fail messages. Others will just continue on.  Clear Log button will reset this display to its original state. |

## Example

This may look complicated at first. But an Application can be created with just a few button clicks. Here’s an example with 5…



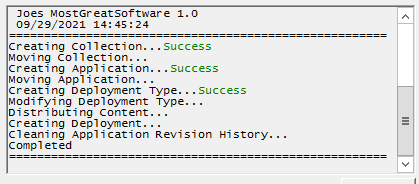
1. Enter package name.
2. Select Script.

* *In this example we are using the PowerShell Application Deployment Toolkit (which is a great free tool) for our package. Install and Uninstall commands are set automatically, imported from the XML.*

1. Within the package we have an MSI. Browse to this to import the Product Code for Detection.

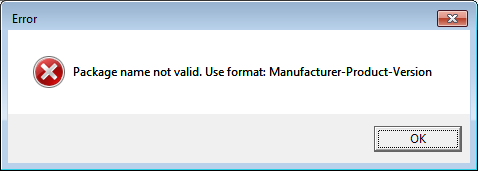
* *I know we could have chosen MSI for the Deployment Type. But in my experiences, most packages (even MSI packages) are called within some sort of wrapper, in this case PSADT. So, I feel that this is the best example.*

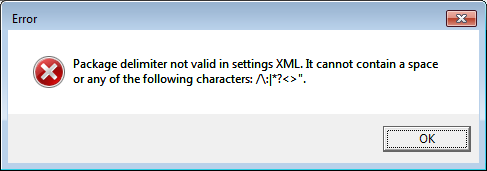
1. Here we chose to only import an icon.
2. Create.
3. Output is logged as the tool runs.



## Troubleshooting

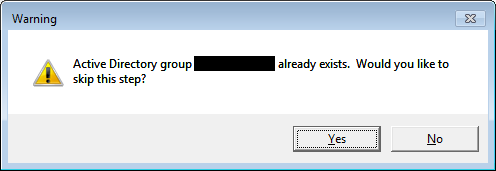
Additional validation checks are in place to check user input. If any fail, an error message will appear on the screen. Such as.





If a step errs during creation of the App, that error should also be displayed in a message window with the full error details.

Some pre-validation checks are also performed at various steps, and if that feature already exists there will be a prompt to Skip or exit. This allows the packager to continue using the tool even if some steps were already completed. Or cancel out if it was unintentional.



Additional troubleshooting may be done by running the tool in debug mode. This is by calling the ps1 with the switch “Dbug” (.\MECMAddAppTool.ps1 -Dbug). This simply keeps the PS console open during run so that any console output is visible, which may include errors.