# MECM AddApp Tool

Version: 2.13.1

Author: Joel Chettle

As of: 4/3/2023

Contents

[MECM AddApp Tool 1](#_Toc113368773)

[Description 1](#_Toc113368774)

[Requirements 1](#_Toc113368775)

[How-to Launch 2](#_Toc113368776)

[Configuration 3](#_Toc113368777)

[How-to Use 4](#_Toc113368778)

[Example 8](#_Toc113368779)

[Troubleshooting 9](#_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 MECM for a package by providing minimal input.

## Requirements

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

**Files Required:**

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

**Files Included but Optional:**

* MECMAddAppTool.bat
* AppIcon.ico
* Changelog.txt
* Debug.bat
* LICENSE.txt

|  |  |
| --- | --- |
|  |  |

## How-to Launch

To run the tool, simply use the bat file. It is just the command to launch powershell.exe and pass in the script file. Alternatively, you can right-click the ps1 file and select “Run with PowerShell”.

The tool must be run with an account that has access to make modifications to the relevant areas of both Active Directory and MECM.

It may take a moment before the interface loads. This is normal.

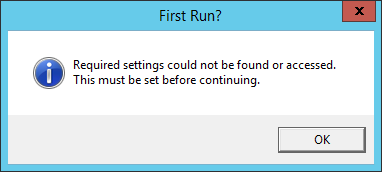
If you have any issues running the tool; for instance, if nothing happens at all, it may be due to Execution Policy. This may be managed in your environment. For more information, please research “PowerShell Execution Policy”. You may also run using the Debug.bat file to confirm, as that will leave the console window open to view any error messages.

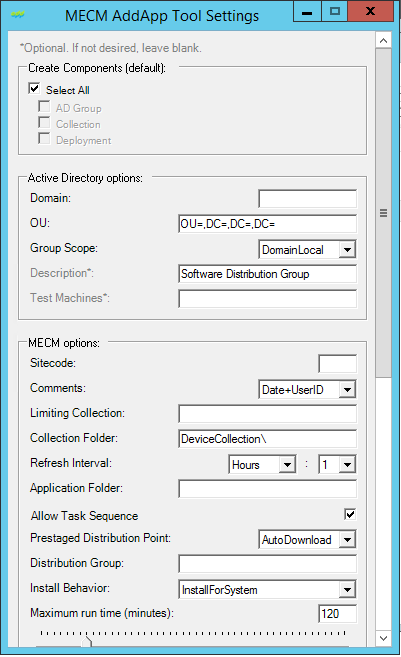
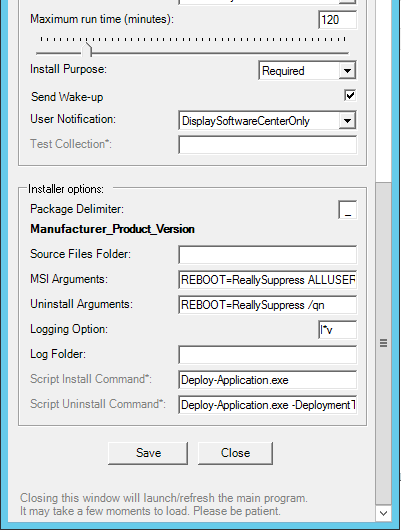
The tool used to be provided as a compiled EXE (with the use of PS2EXE-GUI by Markus Scholtes (a rework of PS2EXE by Igor Karstein)). Unfortunately, this regularly would get flagged by anti-virus software as a false positive. So, I have decided against using this in any current release.

## Configuration

All custom settings are stored in a settings XML. This is so the (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 Resources directory relevant to the ps1 if you have write-access there. Otherwise, it will write to %appdata%.

It is possible to set the xml with an admin account, and then subsequent runnings with read-only accounts can launch the tool, as they would only need to read the xml settings thereafter.



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

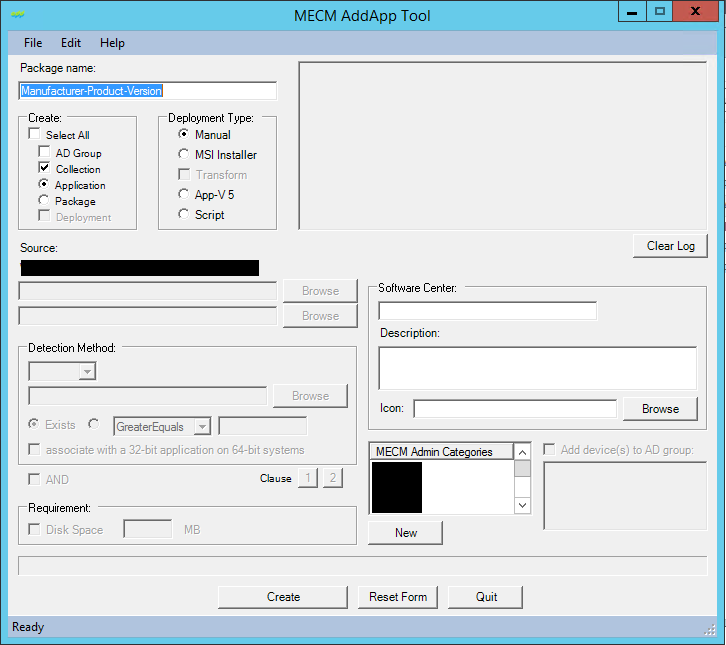
Enter in data/make selections for each required field, then hit “Save”. After it has confirmed “Saved!” you can hit “Close”, and the main form will open.

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 Edit > Settings.

## How-to Use

Interface:

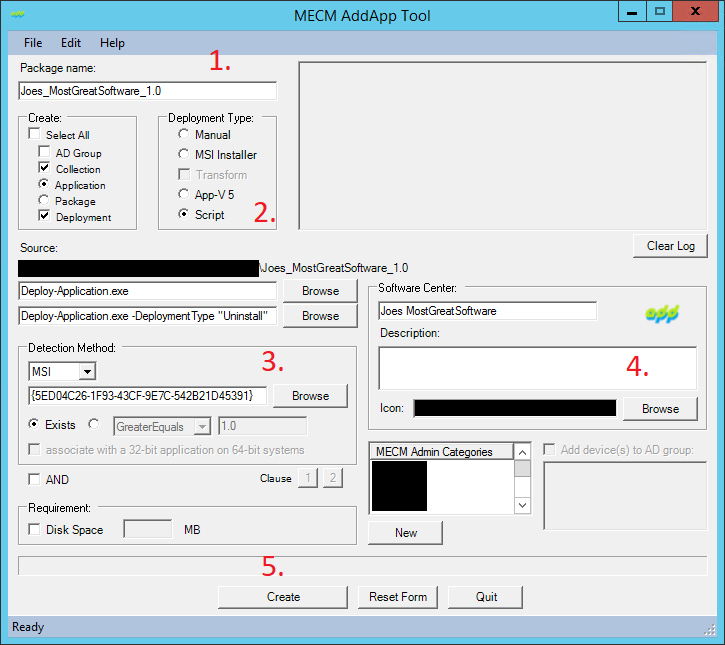


Breakdown:

|  |  |
| --- | --- |
|  | 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. |
|  | 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 coincides with Detection Method. If not selected, detection will use just the single clause defined above. If AND is selected, this allows you to create 2 clauses that will be joined with an “And” condition. The 1 and 2 Clause buttons allow you to swap between these, with the highlighted button being the current Clause displayed.  An OR condition is planned, but currently not implemented. |
|  | Fields used for Software Center appearance.  The first text field is for Localized Application Name. It generates by default with the Product name. But can be modified.  Description and Icon can also be set.  Icon file can be a .ico or .exe. It cannot use .dll’s at this time. Browse allows you to navigate to the file. Once selected, you will see the image of the icon appear on screen.  Note: 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 desired, you can set a Requirement for the Application for Free Disk Space (for the System Drive). Check the box and enter in a value in MB’s.  If not selected, it will not create a Requirement. |
|  | This displays current Admin categories pulled from MECM. Any or multiple can be selected for the App.  New button dynamically creates a new category in MECM and will display it at the bottom of the list. |
|  | If you wish to add machines to the newly created AD group, check the box and enter them in the text field.  If TestMachines 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 excessive at first. But realize that many fields are populated for you. So, an Application can be created with just a few button-clicks. Here’s an example with 5 easy steps…



1. Entered package name.
2. Selected Script.

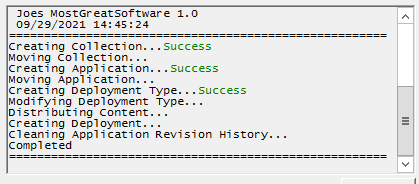
* *In this example we are using the PowerShell Application Deployment Toolkit for our package. If you are not familiar with this, I recommend looking it up. It’s not important, just to reference…*
* ***Install and Uninstall commands are set automatically, imported from the XML.***

1. Within the package I have an MSI. Browsed to this to import the Product Code for Detection.

* *I know we could have chosen MSI for the Deployment Type. But I like demonstrating it this way.*

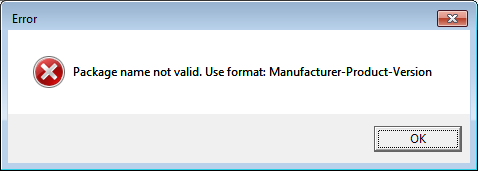
1. Here I chose to import an icon.
2. Create.

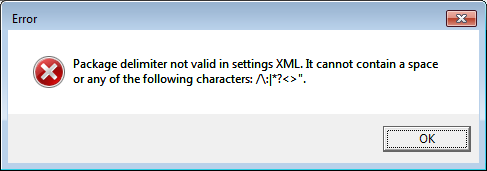
* ***That’s it!***
* *Output is logged as the tool runs.*



## Troubleshooting

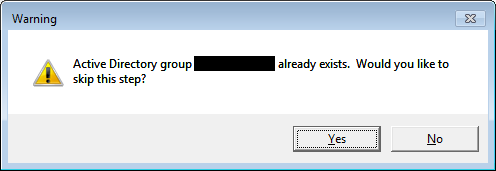
Additional validation checks are in place to check user input. If any fail, an error message will appear on the screen. Such as.





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). Or run Resources\Debug.bat. This simply keeps the PS console open during run so that any output is visible, which may include errors.

\*Some screenshots have sections blacked out to protect the specific information of the sample environment.