Onevinn AB

Web Service for Configuration Manager (free)

Installation and user's manual

onevinn

CONTENTS

1. VERSION

Version	Author	Date	Remark
1.0	Johan Schrewelius	2016-10-17	Document Created
1.2	Johan Schrewelius	2017-08-28	Added support for Driver Manager

2. DISCLAIMER

Onevinn Web Service for Configuration Manager has been tested on SCCM 2012 R2, R2 SP1 and Configuration Manager Current Branch (1606). It is not designed for bulk operations, avoid calling the methods from outside a Configuration Manager Task Sequence; it will probably do the job; but as stated, it's not designed for it.

The free version has not been tested in an SCCM hierarchy, nor in a multi domain forest. Should the demand for these capabilities arise, please contact Onevinn AB for an adapted version!

3. DESCRIPTION

Onevinn Web Services for Configuration Manager is intended for typical TS Actions such as adding or removing a computer from/to Collections and AD-Groups during deployment. It includes methods for retrieving AD-Group memberships (applications) on re-install, as well as a couple of workarounds for unappreciated SCCM behavior (no Policy for this computer, and faulty merging to discovered objects).

Onevinn Web Services for Configuration Manager consists of one webservice, OnevinnWS, typically hosted on IIS on a Primary Site Server, and several PowerShell sample scripts, used during Operating System Deployment.

The advantages of this approach, as opposite of running equivalent stand-alone scripts in Task Sequences are obvious – only port 80/443 will be needed, avoiding firewall issues, it will also isolate the permissions needed to the account running the service's Application Pool.

4. SERVICE ACCOUNTS

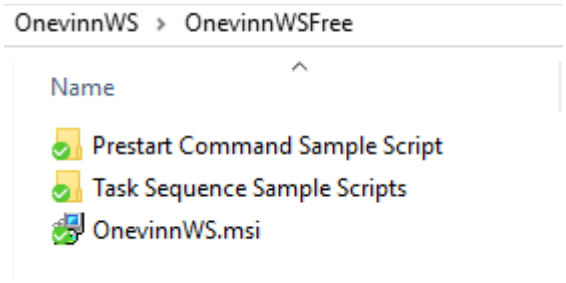
Prior to installation two service accounts needs to be created, these are:

Account	Usage	Permissions
Domain**SVC_OVWS**	IIS Application Pool Identity	**AD**: Add/remove computers to groups, move computers between OU's. **SCCM**: Operations Administrator
Domain**SVC_CMTSAction**	Authorized to Call service	IIS Authorization Rule - Allow. Automatically added during setup. *Make sure these IIS roles are installed.*

5. MEDIA

Presuming that **OnevinnWSFree.zip** has already been downloaded.

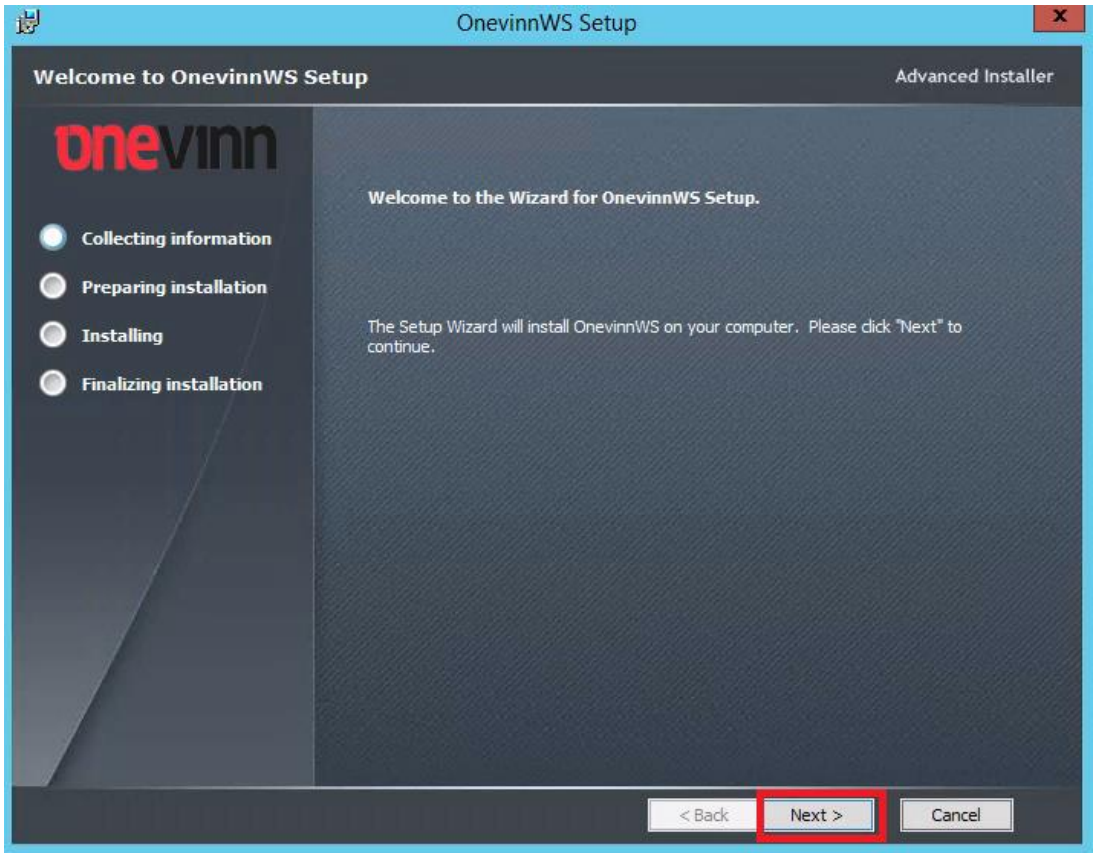
Unpack the archive, you should then have a folder looking like:



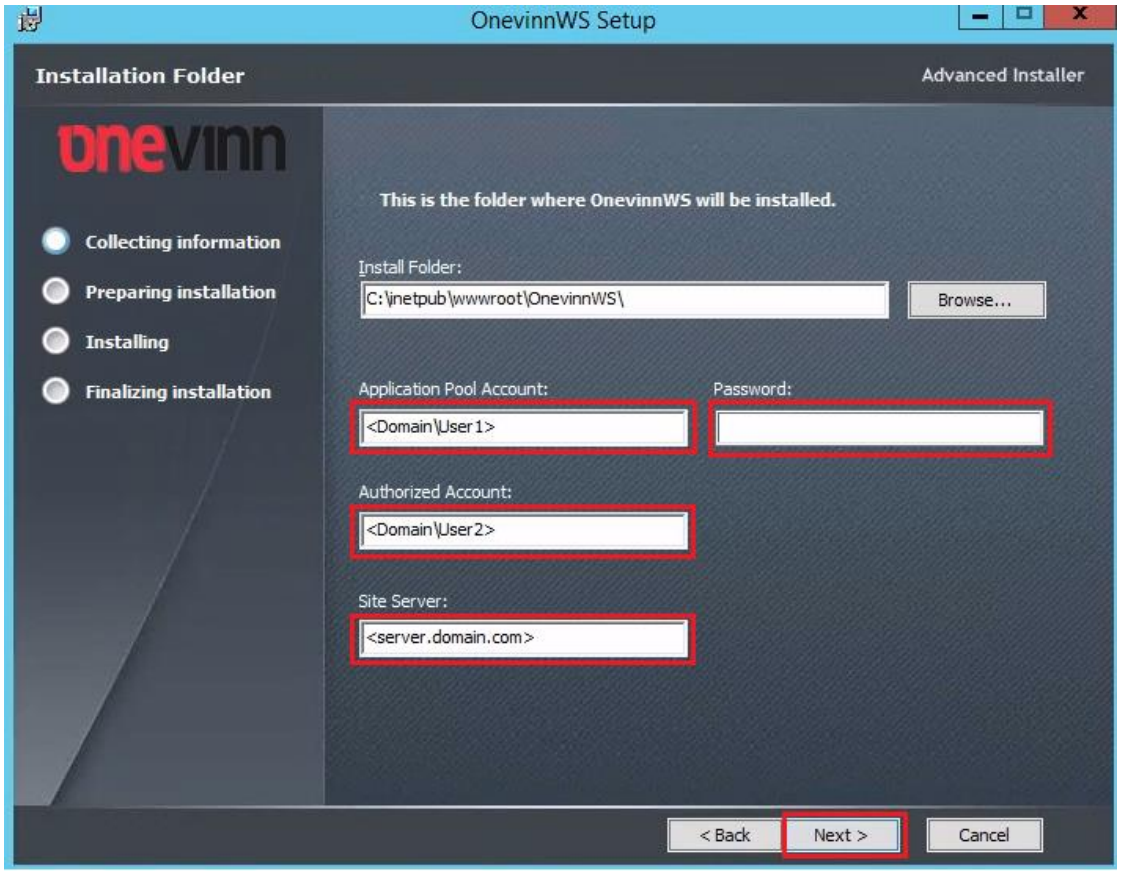
6. INSTALLATION

Copy the installation file, **OnevinnWS.msi,** to your Primary Site server and double click it to start the installation.

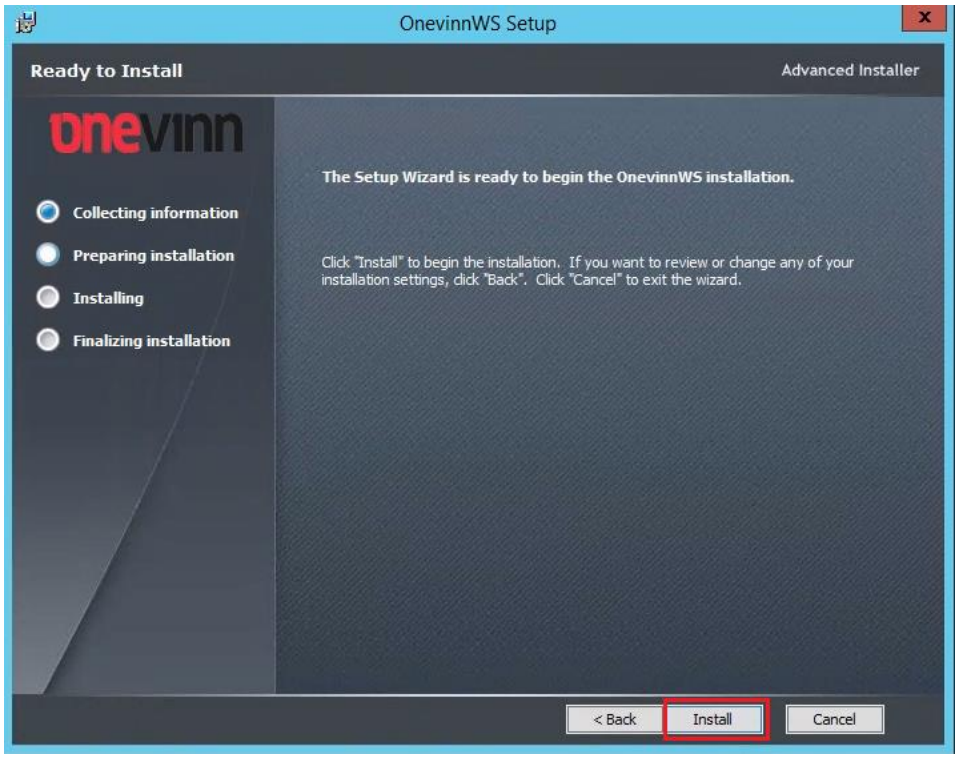
The installer starts, click **Next**.



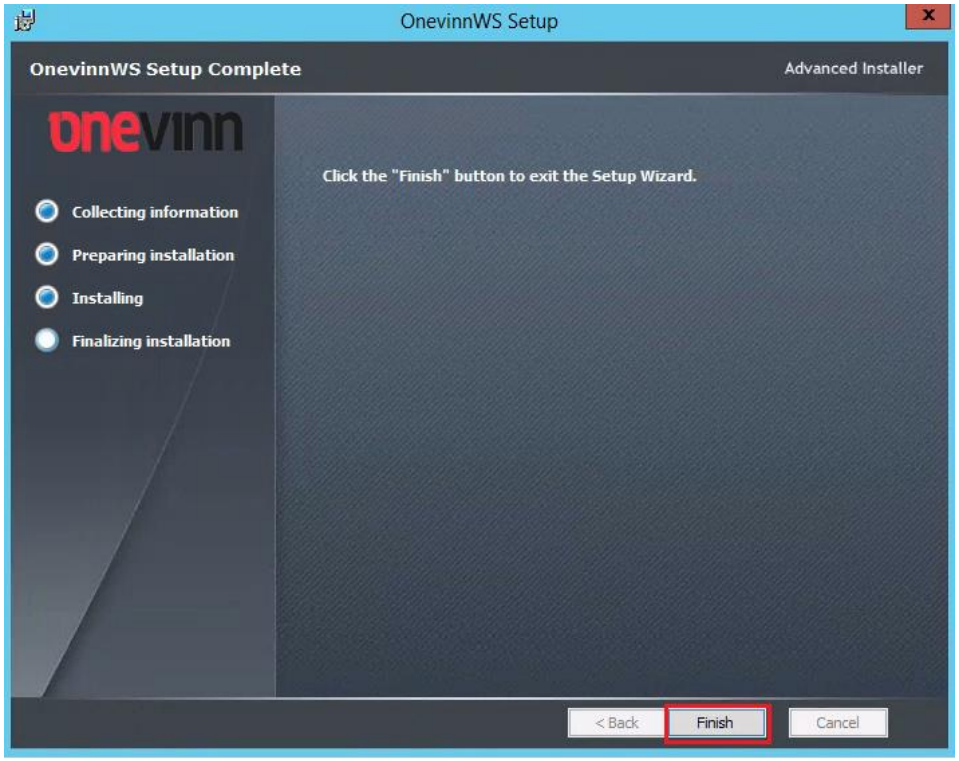
Fill in the accounts, password and the FQDN of the Site Server. **Click Next**.



Click Install to install OnevinnWS.



Click Finish to exit the installer.

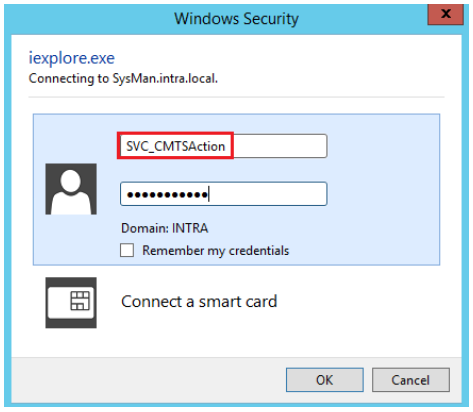


7. METHODS AND TESTS

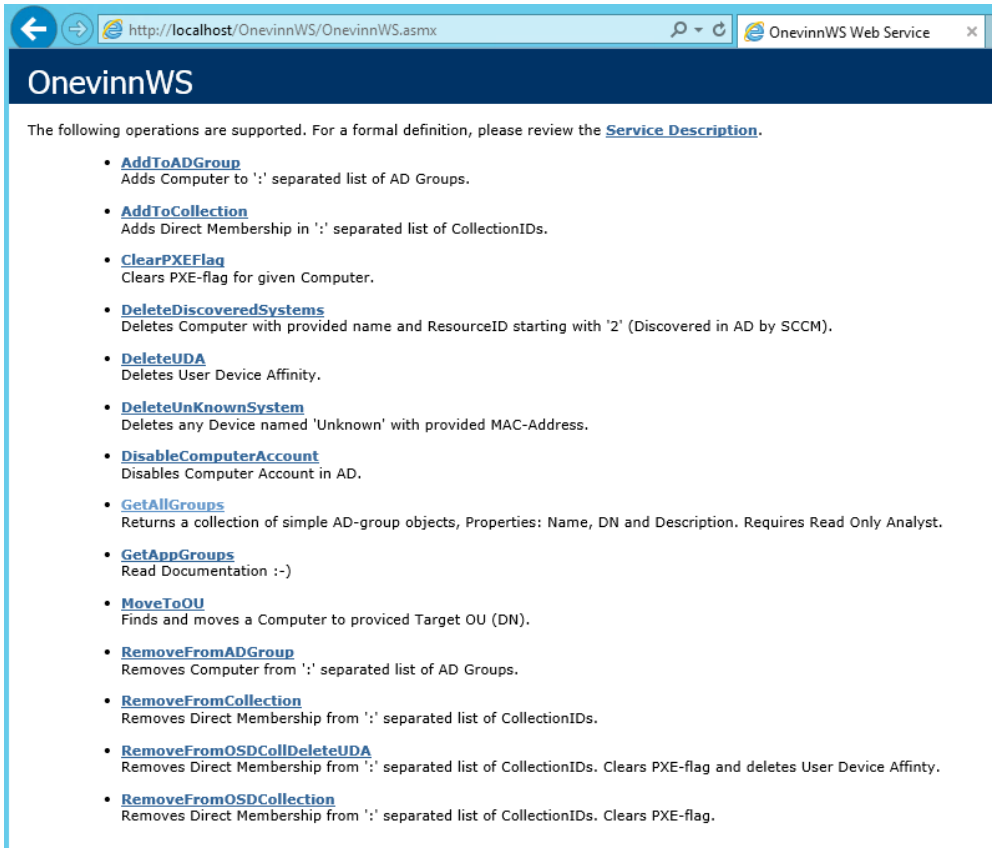
Open up Internet Explorer on the Primary Site Server and browse to:

http://localhost/OnevinnWS/OnevinnWS.asmx

A logon box will appear – the only account allowed to logon is the service account **SVC_CMTSAction**:

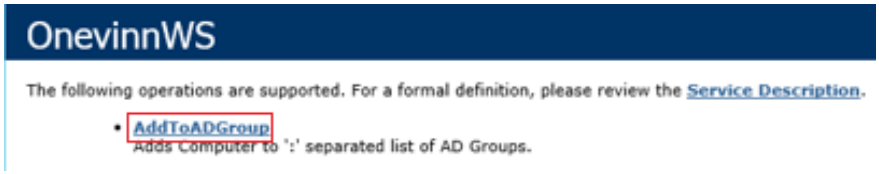


Once logged on all **OnevinnWS** methods will be revealed:

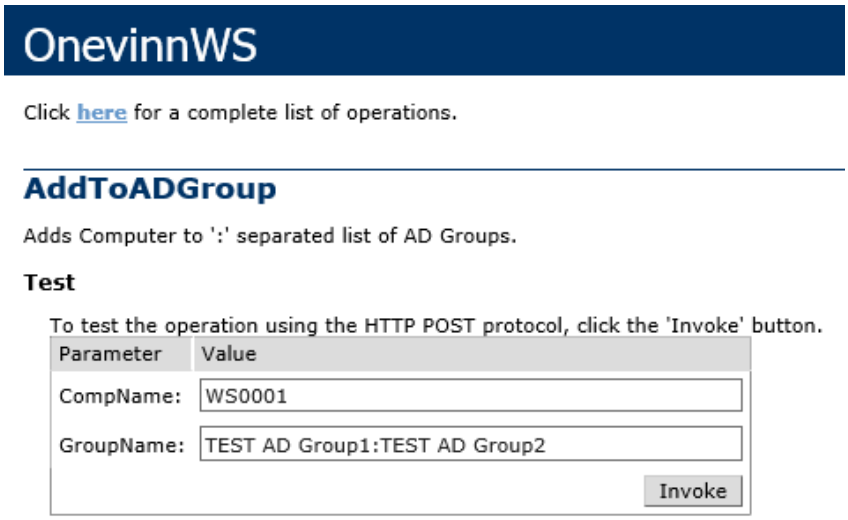


From this page all methods can be tested, for example:

Press “**AddToADGroup**”:



On the next page the method parameters should be filled in:



This would, provided that the account that runs the Application Pool has been delegated the necessary permissions, add computer “WS00001” to two AD-Groups.

Same syntax is used for removing from groups as well as when dealing with SCCM Collections, except the latter requires Collection IDs instead of names.

8. SAMPLE SCRIPTS

The free edition of OnevinnWS is accompanied by several Sample scripts exemplifying how to call the different methods from a SCCM Task Sequence. This guide assumes MDT integration, if MDT is not used TS-Variables will have to be created and set using other means.

The scripts expect the following TS-variables to be present and set during deployment:

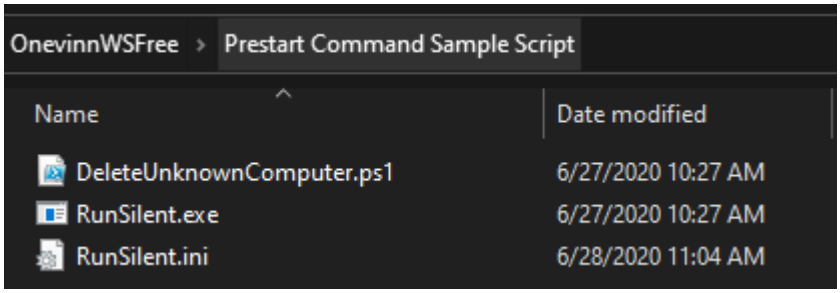
OSDComputerName – NetBios name of computer.

MachineObjectOU – Destination OU for successfully deployed computer.

All TS-Scripts are run as "**Run Command**" steps, which allows parameters the built in "Run Powershelll Script" step doesn't support.

8.1. Prestart Command Sample Script

The folder "**Prestart Command Sample Script**" contains three files:



These are not intended for the TS itself but to be included in the boot-image and run as a Prestart-command.

On line 8-10 in "**DeleteUnknownComputer.ps1**" and "**SetSystemTime.ps1**" configure:

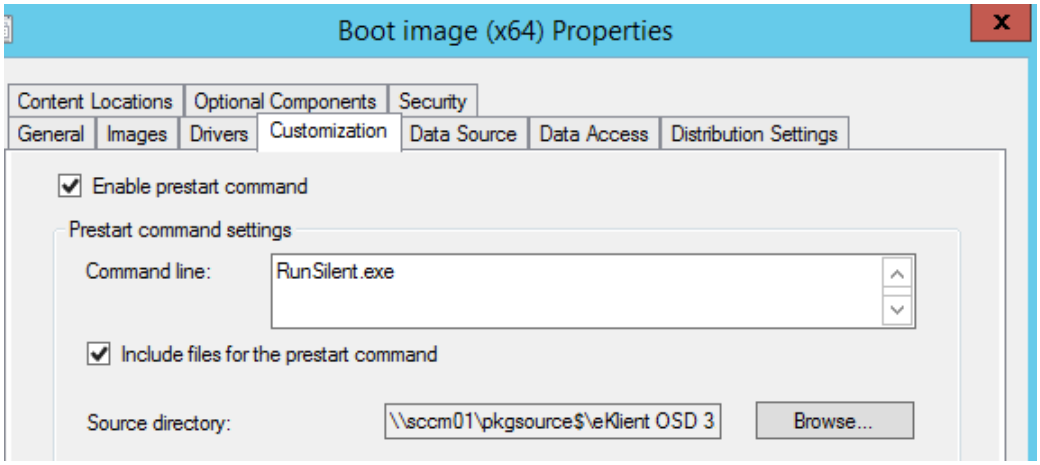
```
[string]$URI = "http://<server.domain.com>/onevinnWS/onevinnws.asmx"
[string]$UsrName = "Domain\SVC_CMTSAction"
[string]$UsrPW = "<Password>"
```

Account to use is of course the only one with permissions on OnevinnWS, **SVC_CMTSAction**.

The purpose of "**DeleteUnknownComputer.ps1"** is to delete any matching "**Unknown**" computer in SCCM by sending the MAC-address of the computer to OnevinnWS, which in turn will delete the possibly existing object from SCCM, making the machine properly unknown, thus allowing it to fetch a deployment policy.

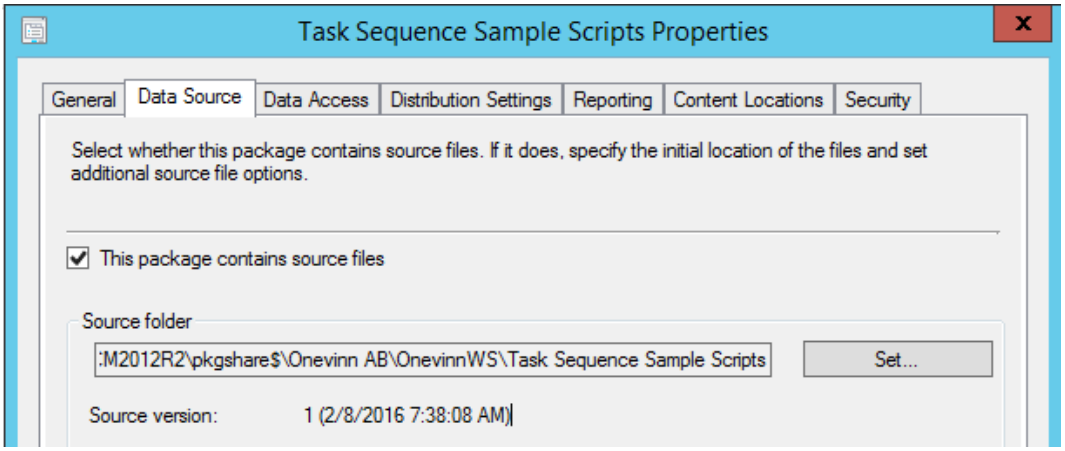
Customize your boot-image by Including the three files as Source directory and run "**RunSilent.exe**" as Prestart command.

"**RunSilent.exe**" will execute the scripts according to information in RunSilent.ini, of course it's possible to add further commands to the ini-file, syntax is self-explanatory. RunSilent.exe suppresses the annoying Powershell Blue Box.



8.2. Task Sequence Sample Scripts

To make the scripts accessible from Task Sequence create a SCCM Package using the “Task Sequence Sample Scripts” as content source. Do not create any programs for the package.



Distribute the package to your Distribution Point(s).

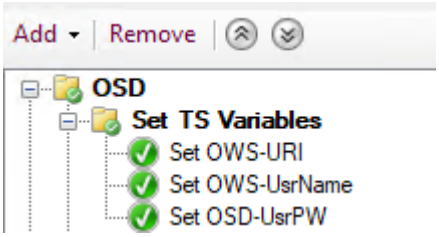
8.3. Configuration TS Variables

In an earlier version configuration was made in clear text in the Configuration.ps1 file, this is no longer the case. Since this service was first released (2016) it has become possible to use hidden TS variables instead, a significantly better solution.

OWS-URL

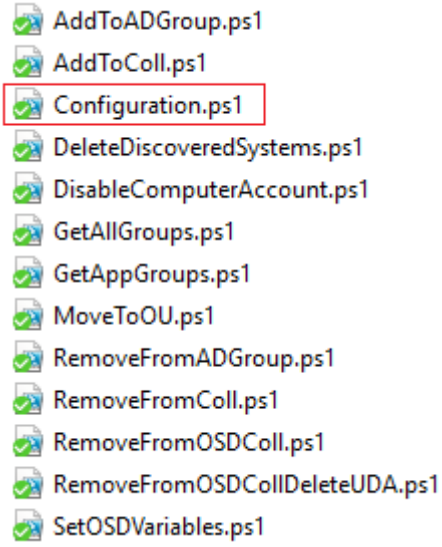
OWS-UsrName

OSD-UsrPW



8.3.1. Configuration.ps1

Task Sequence Sample Scripts:



Configuration.ps1 contains the necessary configuration to retrieve AD group memberships for applications.

```
# Configuration for AD group filtering (Applications) #################

[string]$SiteCode = "<SiteCode>"
[string]$Prefix = "A."
[string]$InstallSuffix = ".i"
[string]$UnInstallSuffix = ".u"
[Bool]$RemovePrefix = $true
[bool]$IncludeComputerGroups = $true
[bool]$IncludeUserApps = $false
```

More about this later.

8.3.2. AddToADGroup.ps1

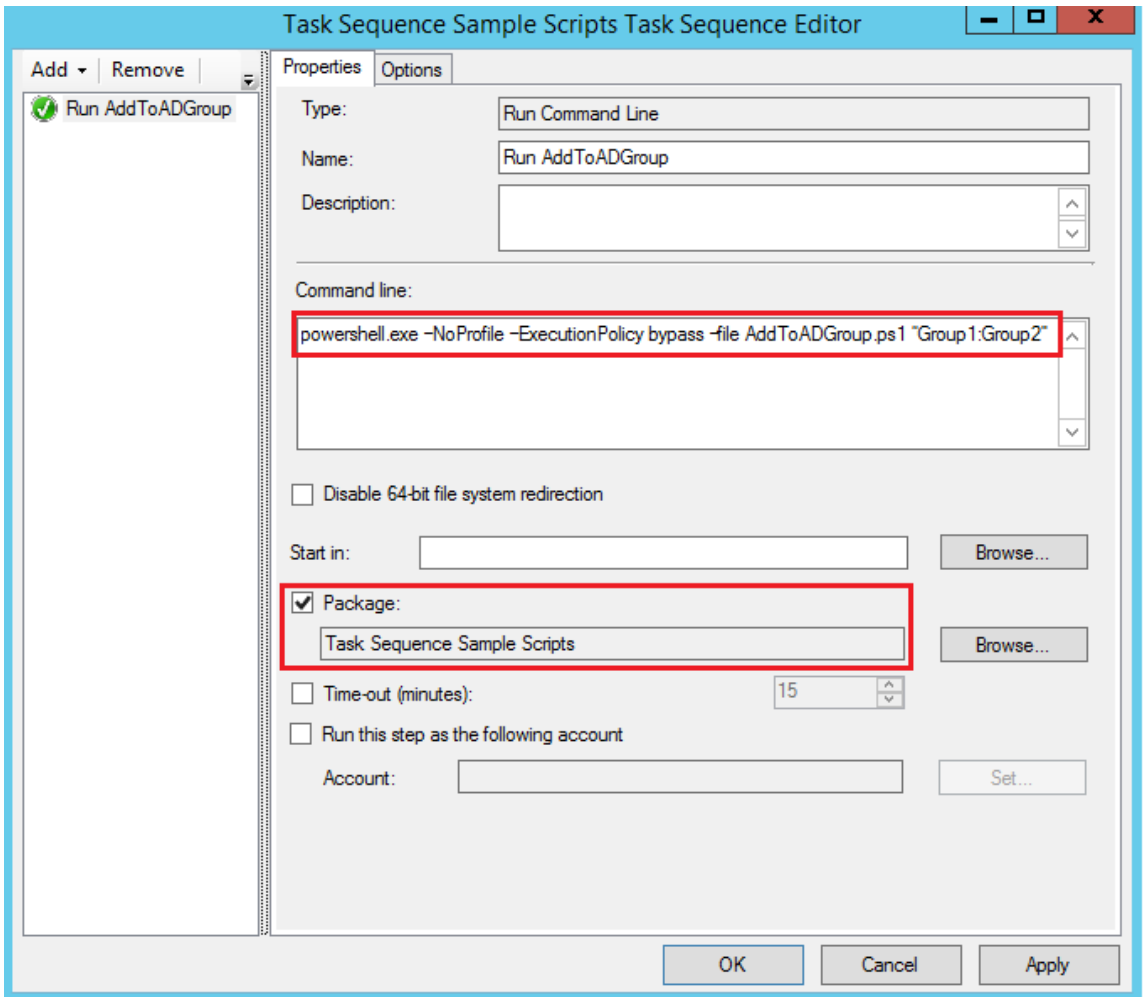
Description: Adds a computer to one or many AD Groups.

Command:

powershell.exe -NoProfile -ExecutionPolicy bypass -file AddToADGroup.ps1 "Group1:Group2"

Arguments: *Yes*

Example:



8.3.3. AddToColl.ps1

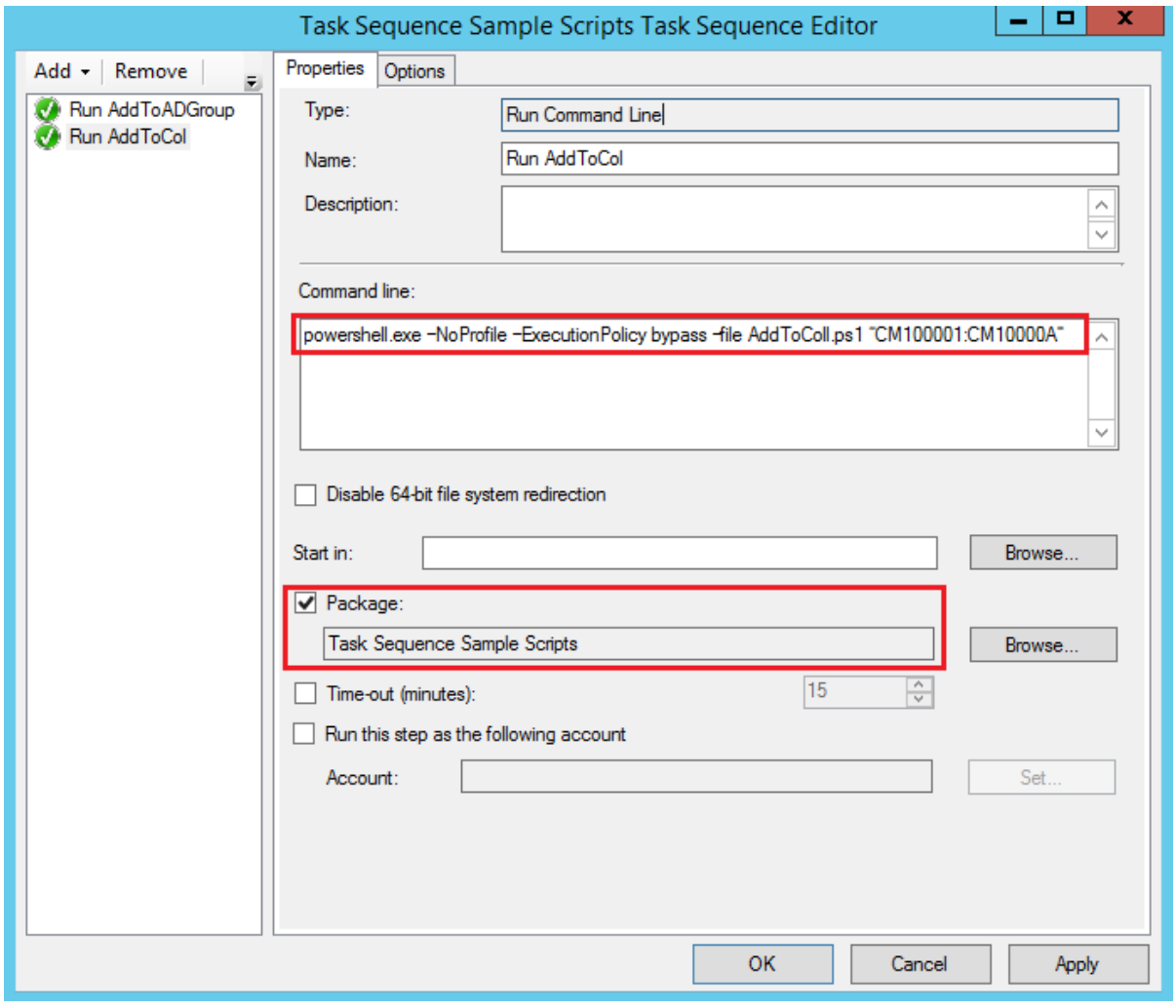
Description: Adds a computer to one or many Collection.

Command:

powershell.exe -NoProfile -ExecutionPolicy bypass -file AddToColl.ps1 "CM100001:CM10000A"

Arguments: *Yes*

Example:



8.3.4. RemoveFromADGroup.ps1

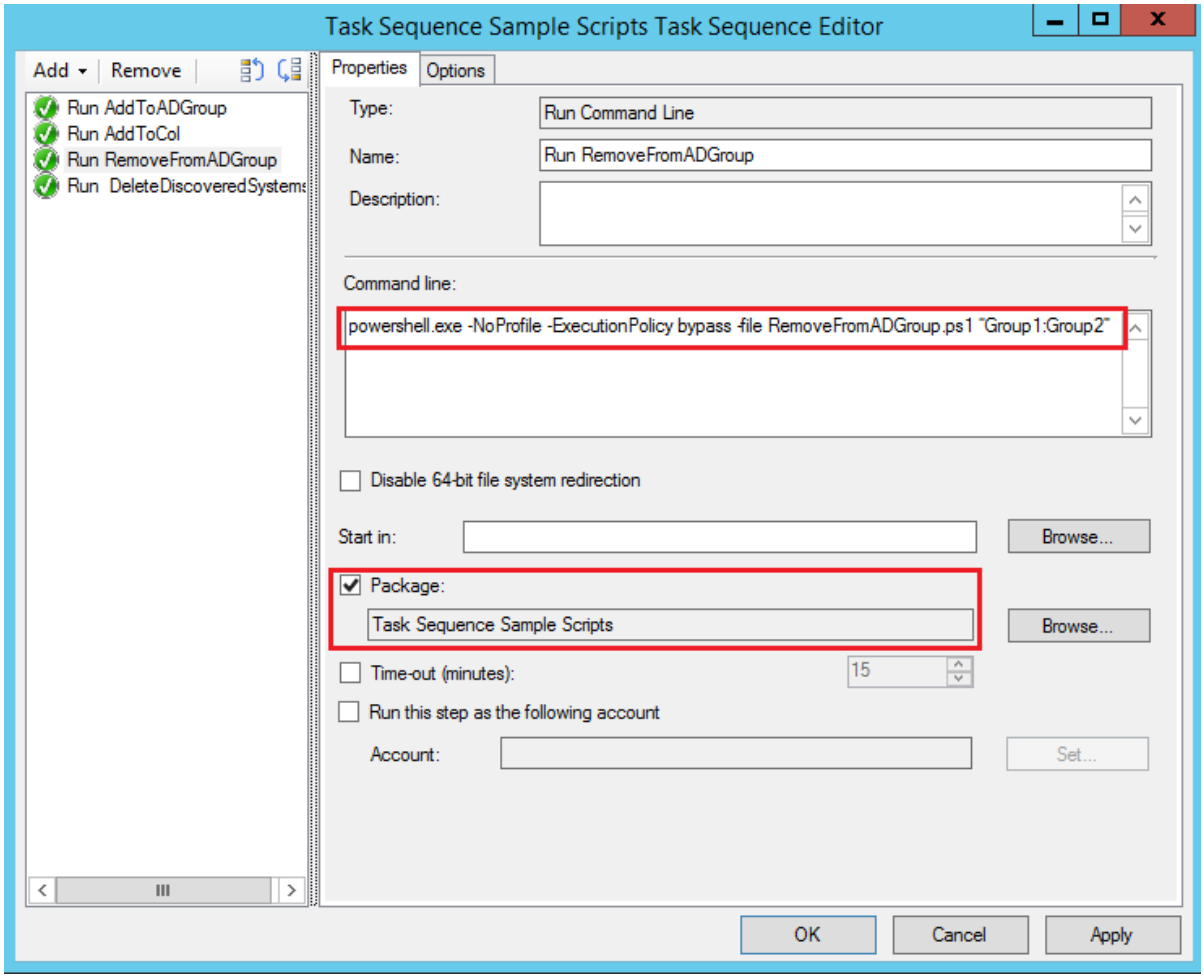
Description: Removes a computer from one or many Collection.

Command:

powershell.exe -NoProfile -ExecutionPolicy bypass -file RemoveFromADGroup.ps1 "Group1:Group2"

Arguments: *Yes*

Example:



8.3.5. Remove from Collection x3

There are three scripts available for removing a computer from one or many Collections.

Description:

RemoveFromColl.ps1 – Removes computer from Collections(s).

RemoveFromOSDColl.ps1 – Removes computer from Collections(s) and clears PXE-flag.

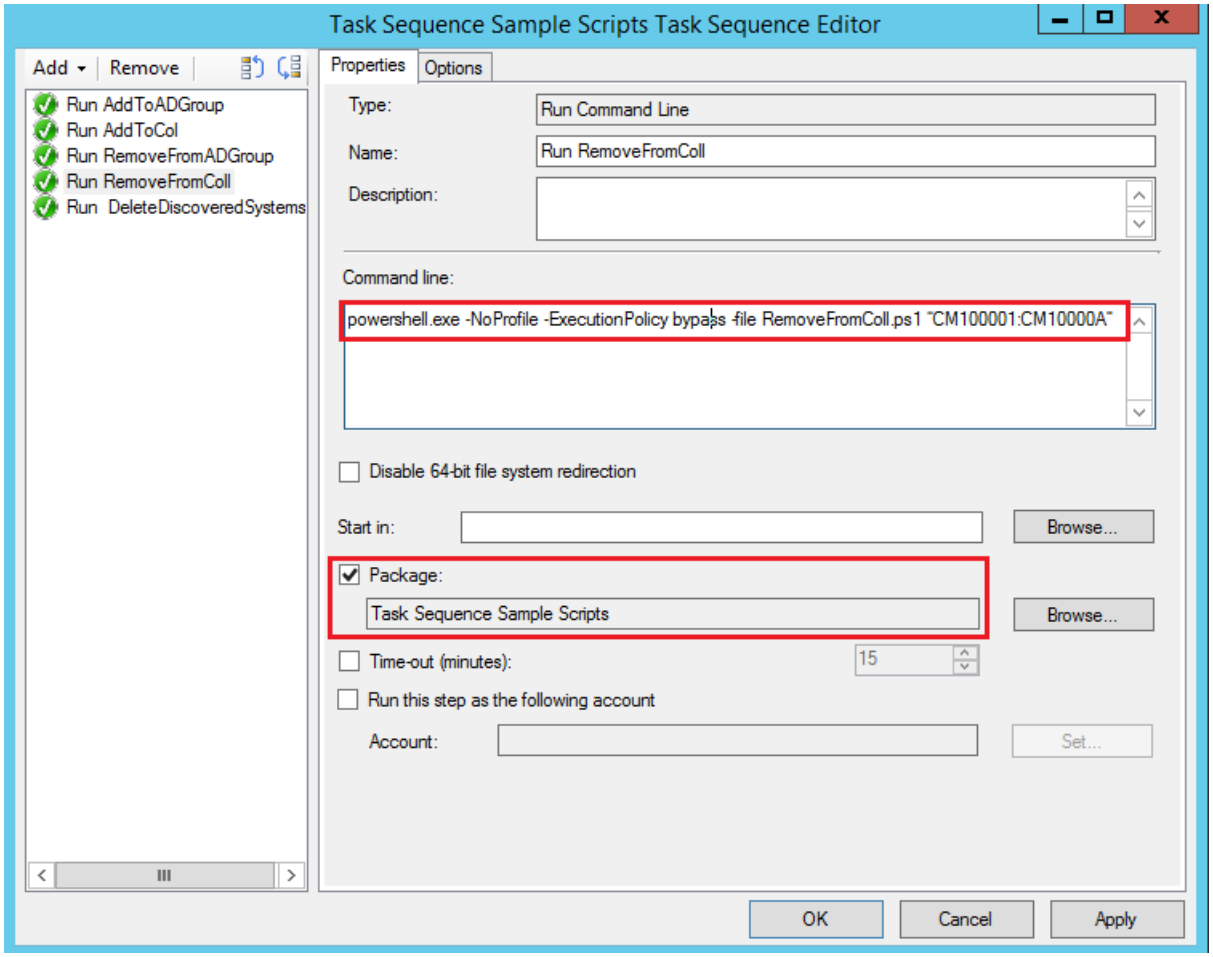
RemoveFromOSDCollDeleteUDA.ps1 – Removes computer from Collections(s), clears PXE-flag and deletes User Device Affinity.

Command:

powershell.exe -NoProfile -ExecutionPolicy bypass -file <ScriptName>.ps1 "CM100001:CM10000A"

Arguments: *Yes*

Example:



8.3.6. DeleteDiscoveredSystems.ps1

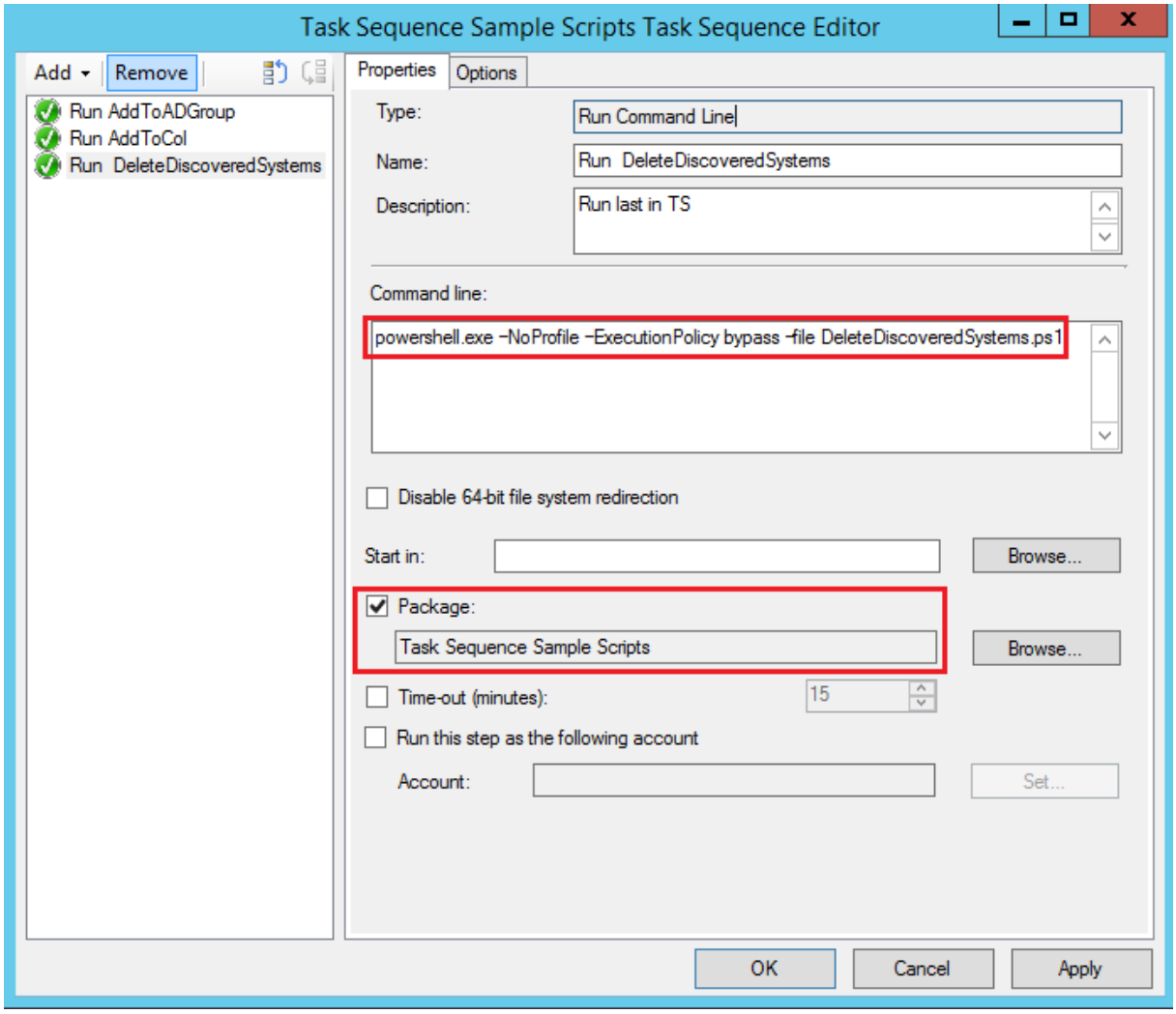
Description: Deletes doublet computers in SCCM (Discovered) allowing an 'Unknown' computer to merge with correct object after TS, **should be run as last step in TS**.

Command:

powershell.exe -NoProfile -ExecutionPolicy bypass -file DeleteDiscoveredSystems.ps1

Arguments: *No*

Example:



8.3.7. MoveToOU.ps1

Description: Moves a computer to a different OU. Utilizes TS-Variable "**MachineObjectOU**", that has to be set prior to calling the method.

Command:

powershell.exe -NoProfile -ExecutionPolicy bypass -file MoveToOU.ps1

Arguments: *No*

Example:

Same as previously described scripts that doesn't require argument, ex. "DeleteDiscoveredSystems".

8.3.8. DisableComputerAccount.ps1

Description: Intended to run during Error handling in TS, disables the computer account in Active Directory, preventing logon and use of a computer that failed deployment.

Command:

powershell.exe -NoProfile -ExecutionPolicy bypass -file DisableComputerAccount.ps1

Arguments: *No*

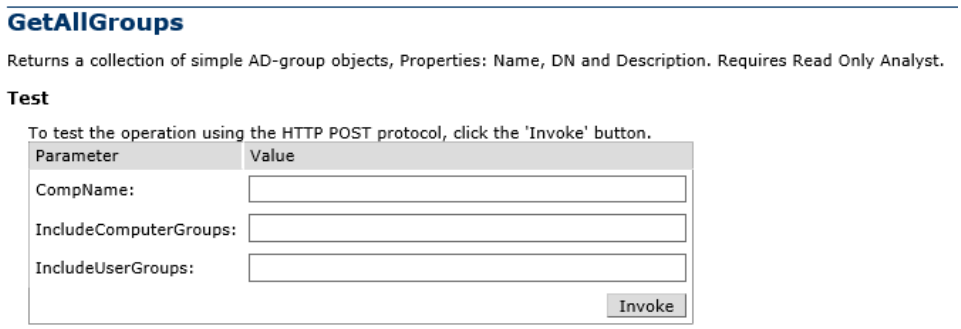
Example:

Same as previously described scripts that doesn't require argument, ex. "DeleteDiscoveredSystems".

8.3.9. GetAllGroups.ps1 – Advanced

Description: Retrieves AD-Group membership (memberof) information for the deployed computer. OnevinnWS will return a list of objects containing, **Name, distinguishedName and description of each group**. By iterating over that list one can decide to take certain actions, the sample script demonstrates how to create TS-Variables for application installation.

OnevinnWS will include computer memberships and memberships for Pimary User based on what arguments are passed (Set in configuration.ps1).



Command:

powershell.exe -NoProfile -ExecutionPolicy bypass -file GetAllGroups.ps1

Arguments: *No*

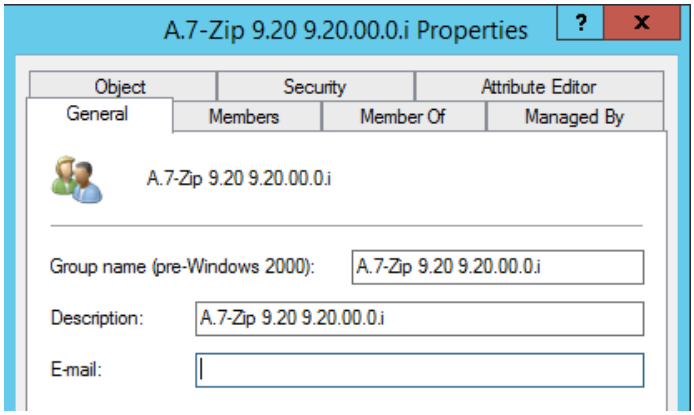
Example:

This script is considered to be “Advanced” and the usage is limited only by innovational skills of the user.

8.3.10. GetAppGroupsAndLog.ps1 – Advanced

Description: Like "GetAllGroups.ps1", except that some logic can be performed server-side. OnevinnWS will filter the groups by comparing the content of each groups Description attribute to the settings in the second section of "Configuration.ps1".

This function will return the value of the groups Description attribute, stripped, or not, of prefix/suffix. For example:




```
[string]$Prefix = "A."
[string]$InstallSuffix = ".i"
[string]$UnInstallSuffix = ".u"
[Bool]$RemovePrefix = $true
[bool]$IncludeComputerGroups = $true
[bool]$IncludeUserApps = $false
```

Would result in the sample script creating a TS-Variable:

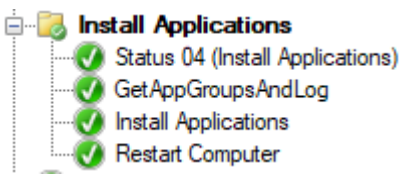
COALESCEDAPPS01 = "7-Zip 9.20 9.20.00.0"

Command:

powershell.exe -NoProfile -ExecutionPolicy bypass -file GetAppGroups.ps1

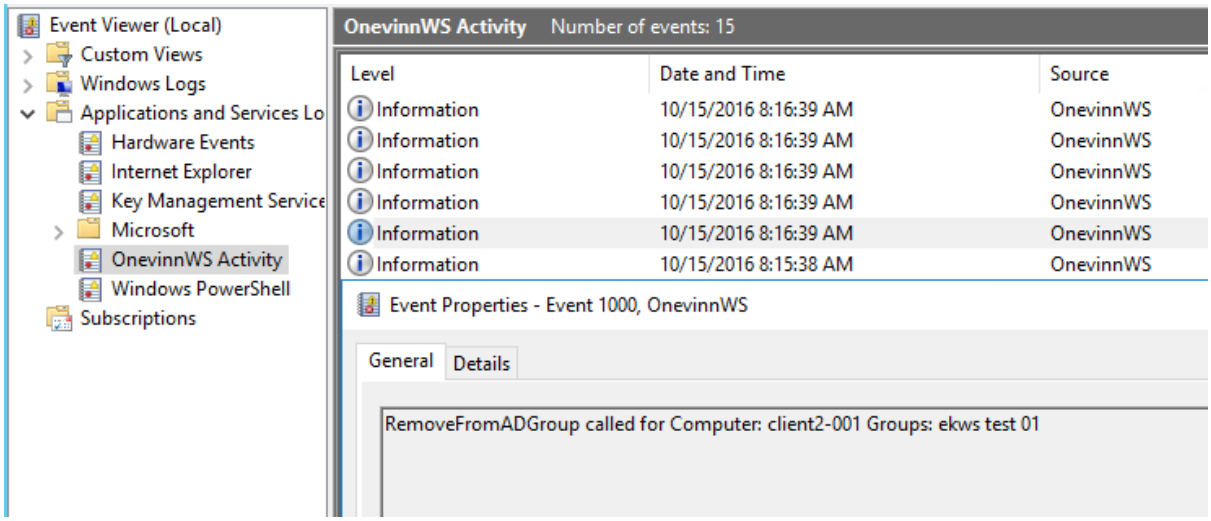
Arguments: *No*

Example:



9. LOGGING

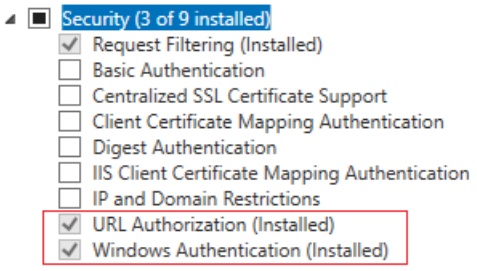
OnevinnWS logs more or less all activity to its own log under **Applications and Services Logs**, making it easy to find eventual problems.



10. IIS

Should you decide to run OnevinnWS on another server than the PSS (Primary Site Server), IIS will need, **apart from defaults**, a couple of security features. These are:

- URL Authorization
- Windows Authentication



11. ISSUES

Normally the IIS roles above are installed on a PSS that is also an MP or DP – if not, make sure they are added.