



CA Setup Helper- Manual

Prepared by

Andreas Luy

29. Apr. 2025

Version 1.0

anluy@microsoft.com

Table of Contents

Table of Contents

Table of Contents	2
Introduction	3
How to use the ADCS Setup Helper.....	4
Prerequisites	4
Getting started	4
Creating Customer-specific CA Configuration files	4
Running the Script	5
Appendix.....	9
Arguments accepted by the Script.....	9
List of Figures	10
List of Tables	10

Introduction

The ADCS CA Setup Helper provides a convenient way of installing and configuring ADCS by automating installation and applying settings from customer-specific configuration files.

Setup of ADCS based on XML-configuration files is a two-step approach:

Step 1 installs the binaries and creates the certificate requests, while step 2 installs the CA certificate and applies all the necessary configuration. Active Directory-integrated CAs must be rebooted between step 1 and 2 to ensure that Kerberos tokens get refreshed.

Customer configuration files may define one or more Certification Authorities and their configuration.

The customer configuration files are written in the XML Format.

Note: For the future, it is planned to include more PKI-related services as OCSP and NDES.

How to use the ADCS Setup Helper

Prerequisites

- You will need Windows PowerShell Version 4.0 or newer installed on your machine.
- You need to be local administrator to run the script in **"Install"** mode.
- You don't need to have administrative permissions to run the script in **"View"** mode.

Getting started

The base component to setup a CA is the CA configuration XML file. Each configuration file starts with a config node. The config node contains information on the schema file that is used to verify the configuration. Each CA configuration must contain one CA node. The node contains all configuration settings for a single CA.

Creating Customer-specific CA Configuration files

To create customer-specific configuration files, take the sample configuration files that can be found in the "conf" Subdirectory of the Script.

Note: **Never rename or edit the sample configuration files directly.** Make a copy of them and store them separately.


 Sample_Fabrikam Issuing CA 1.xml	11/4/2019 9:06 AM	XML Document	20 KB
 Sample_Fabrikam Root CA.xml	11/4/2019 9:06 AM	XML Document	21 KB

Figure 1: Sample CA Configuration files

Each CA configuration starts typically with the configuration file for the root CA. Create a copy of the file and rename it as desired.

Note: You should use an **XML editor with syntax highlighting** to edit the customer configuration files. You may otherwise have serious trouble properly formatting them. Give **Visual Studio Code** a try!

Note: The CA configuration files are mostly self-explanatory and/or inline commented.

We strongly recommend making yourself familiar with the structure of the configuration files before running this in a customer's environment.

Running the Script

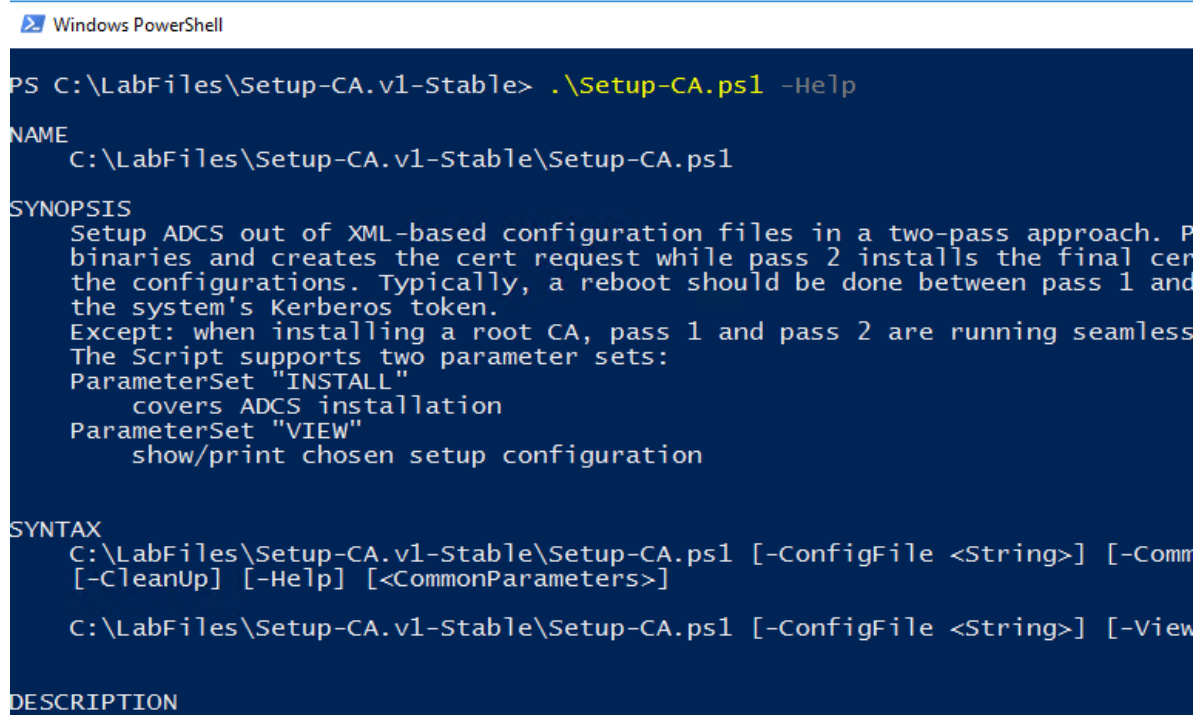
Copy the provided ADCS Setup Helper ZIP file onto the designated new CA and unpack.

Open an elevated Windows PowerShell and navigate to the folder where the ADCS Setup Helper files are stored. You run the script with:

```
.\Setup-CA.ps1
```

If you do specify **-Help** as argument on the command line, it will print the help section of the script.

```
.\Setup-CA.ps1 -Help
```



```
Windows PowerShell
PS C:\LabFiles\Setup-CA.v1-Stable> .\Setup-CA.ps1 -Help
NAME
    C:\LabFiles\Setup-CA.v1-Stable\Setup-CA.ps1
SYNOPSIS
    Setup ADCS out of XML-based configuration files in a two-pass approach. Pass 1 installs the binaries and creates the cert request while pass 2 installs the final cert and does all the configurations. Typically, a reboot should be done between pass 1 and pass 2 to refresh the system's Kerberos token.
    Except: when installing a root CA, pass 1 and pass 2 are running seamless.
    The Script supports two parameter sets:
    ParameterSet "INSTALL"
        covers ADCS installation
    ParameterSet "VIEW"
        show/print chosen setup configuration
SYNTAX
    C:\LabFiles\Setup-CA.v1-Stable\Setup-CA.ps1 [-ConfigFile <String>] [-CommonName <String>] [-CleanUp] [-Help] [<CommonParameters>]
    C:\LabFiles\Setup-CA.v1-Stable\Setup-CA.ps1 [-ConfigFile <String>] [-View]
DESCRIPTION
```

Figure 2: Running the Script with -Help Argument

Note: For a general understanding, we first work with sample data.

Install and configure ADCS out of XML-based configuration files in a two-step approach. Step 1 installs the binaries and creates the cert request while step 2 installs the final cert and is doing all the configurations. Typically, a reboot should be done between step 1 and step 2 to refresh the system's Kerberos token.

Except: when installing a root CA, steps 1 and 2 are running seamless without reboot.

The following is only valid for installing a subordinated CA

After the step 1 run, the script pops out the request file for the CA certificate and asks for a reboot. The request file can be found at the root of the system drive (typically c:\):

```
-----
Configuring the Certification Authority Role
-----
WARNING: The Active Directory Certificate Services installation is incomplete. To complete the installation, use the
request file "C:\csr_Fabrikam_Issuing_CA_1.req" to obtain a certificate from the parent CA. Then, use the Certification
Authority snap-in to install the certificate. To complete this procedure, right-click the node with the name of the
CA, and then click Install CA Certificate. The operation completed successfully. 0x0 (WIN32: 0)

ErrorId      : 398
ErrorString  : The Active Directory Certificate Services installation is incomplete. To complete the installation, use
the request file "C:\csr_Fabrikam_Issuing_CA_1.req" to obtain a certificate from the parent CA. Then,
use the Certification Authority snap-in to install the certificate. To complete this procedure,
right-click the node with the name of the CA, and then click Install CA Certificate. The operation
completed successfully. 0x0 (WIN32: 0)

The CA Service has been installed.
You must now submit the Certificate Signing Request (to be found under C:\csr_Fabrikam_Issuing_CA_1.req) to a Root CA.
Once you have your CA Certificate, name it .\certnew.cer, place it in the same Directory as this script.
Then, run the Script again to finish the Installation.
```

Figure 3: Result of step 1 run of the script. The text includes the CSR file name and location.

Now, you need to transfer the CSR file to your upper tier CA. At your upper tier CA, submit the request file, sign and issue the final certificate. Now export the certificate in DER binary format and get the final certificate back to your system (**we currently don't support PKCS#7 files within the script**).

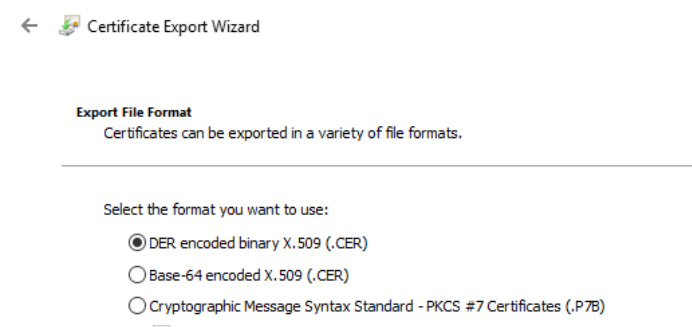


Figure 4: Export the issued certificate in DER encoded binary format.

Save the certificate file in the script's root folder and rename it to "**certnew.cer**".

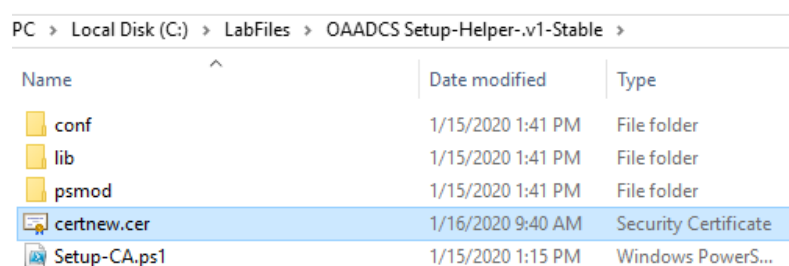


Figure 5: Save the certificate file as "certnew.cer" into the script's root folder.

Before you continue with the step 2 run, ensure you established the appropriate CA certificates chain trust (AIA/root CA trust), especially if you are installing a non-domain joined CA system. For enterprise CAs, ensure you reboot them before continuing.

Note: Ensure you are using the same account for step 2 as you did for step 1.

The Command Line Arguments

ConfigFile

To specify customer configuration files, you can add the **-ConfigFile** argument.

```
.\Setup-CA.ps1 -ConfigFile "Path-to\Customer-CAConfig.xml"
```

Note: If you omit the **-ConfigFile** Argument, a file selection dialogbox will open to select the config file.

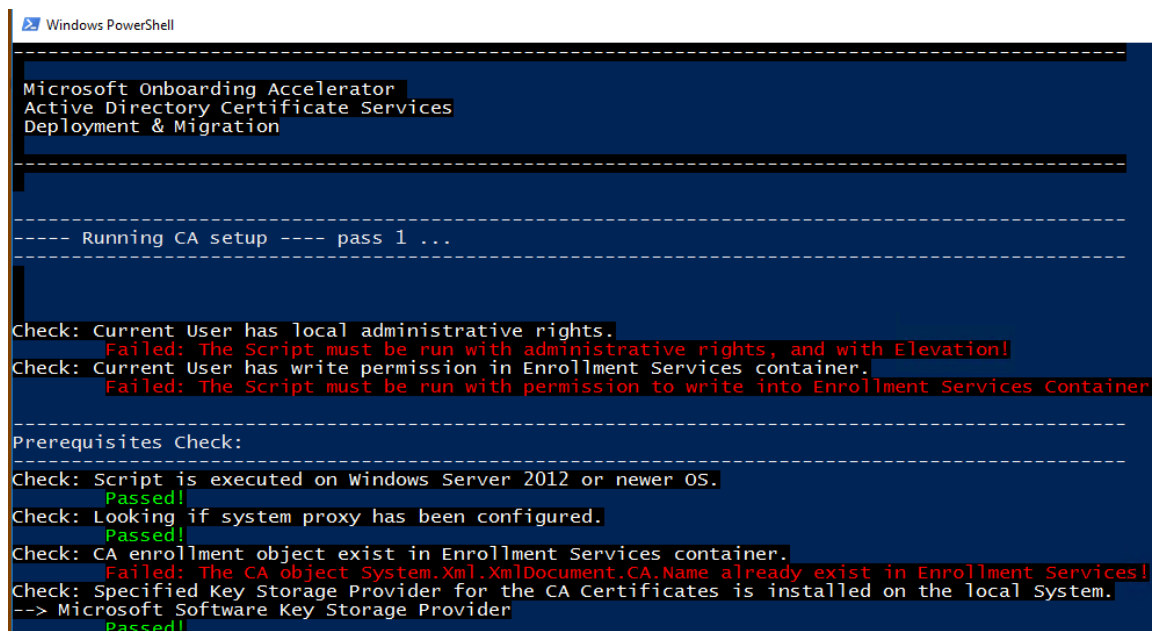
The configfile selection is only important for step 1. During the installation process, the value of "ConfigFile" will be written into the registry and reused for step 2.

Commit

The script will not do anything if there are no additional arguments given. To make the script do something, you will have to specify at least either the **-Commit** or the **-ViewConfig** argument.

```
.\Setup-CA.ps1 -ConfigFile "Path-to\Customer-Config.xml" -ViewConfig  
.\\Setup-CA.ps1 -ConfigFile "Path-to\Customer-Config.xml" -Commit
```

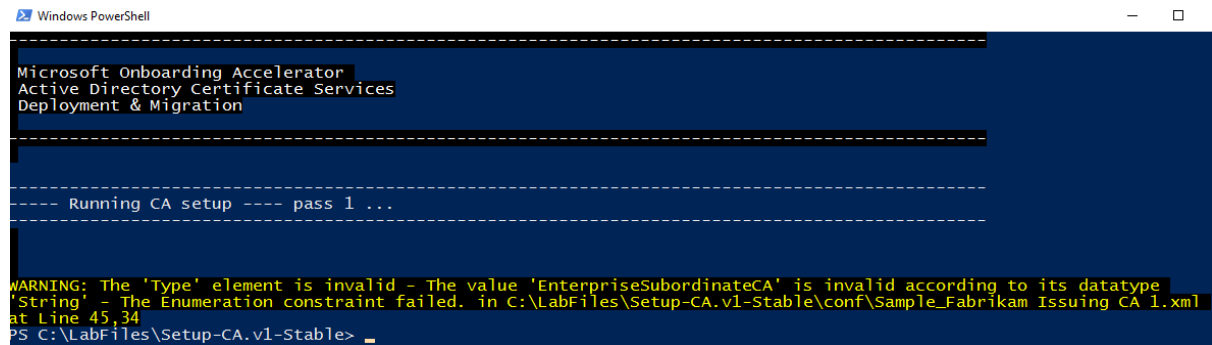
The **-Commit** argument will make the script to install ADCS, but it is recommended to **first run it without the -Commit argument**, it will verify the configuration files and display the configuration and any errors detected.



```
Windows PowerShell  
-----  
Microsoft Onboarding Accelerator  
Active Directory Certificate Services  
Deployment & Migration  
-----  
---- Running CA setup ---- pass 1 ...  
-----  
Check: Current User has local administrative rights.  
Failed: The Script must be run with administrative rights, and with Elevation!  
Check: Current User has write permission in Enrollment Services container.  
Failed: The Script must be run with permission to write into Enrollment Services Container  
-----  
Prerequisites Check:  
-----  
Check: Script is executed on Windows Server 2012 or newer OS.  
Passed!  
Check: Looking if system proxy has been configured.  
Passed!  
Check: CA enrollment object exist in Enrollment Services container.  
Failed: The CA object System.Xml.XmlDocument.CA.Name already exist in Enrollment Services!  
Check: Specified Key Storage Provider for the CA Certificates is installed on the local System.  
--> Microsoft Software Key Storage Provider  
Passed!
```

Figure 6: Script Run without the -Commit Argument, to check for errors

In both cases, the script will verify the input configuration files against XML schemata. If there is an error, the script will halt and display the verification error.



```
Windows PowerShell

Microsoft Onboarding Accelerator
Active Directory Certificate Services
Deployment & Migration

-----
Running CA setup --- pass 1 ...
-----

WARNING: The 'Type' element is invalid - The value 'EnterpriseSubordinateCA' is invalid according to its datatype
'String' - The Enumeration constraint failed. in C:\LabFiles\Setup-CA.v1-Stable\conf\Sample_Fabrikam_Issuing_CA_1.xml
at Line 45,34
PS C:\LabFiles\Setup-CA.v1-Stable>
```

Figure 7: Script Run without the `-Commit` Argument, using sample Data with Errors in the XML File

Therefore, it is recommended to run the script first without the **-Commit** argument and then re-run it with the **-Commit** argument if there were no errors found.

OverwriteConfigInRegistry

During step 1 the used **ConfigFile** and **setup account** are written into the system's registry and reused/validated automatically during step 2. If **-OverwriteConfigInRegistry** switch is defined for step 2, the setup scripts ignore the previously written values in registry and applies the one from the **ConfigFile** (or file selection dialogue) provided with the **-OverwriteConfigInRegistry** argument.

Note: This parameter should be used with care as it allows to mix different **ConfigFiles** and **setup accounts**.

CleanUp

The **-CleanUp** argument is used to clean up the registry from temporary setup status entries done in step 1. Can be used in any step 1 (in step 1 only together with the "**-Commit**" switch) to automatically remove the temporary setup status entries after successful installation. When calling **CleanUp** after successful installation, it removes left over registry entries from ADCS installation.

ViewConfig

ViewConfig let you verify and print a specific CA configuration. ViewConfig will not check for any installation prerequisites and cannot be used with any of the other switches above.

Appendix

Arguments accepted by the Script

Argument	Required	Description
ConfigFile	optional	Specify the Customer Configuration file here. It may either be a relative or an absolute path. If setup is running pass 2, the ConfigFile entered in pass 1 has been written into registry and will be used as default. Only be used when entered together with "OverwriteConfigInRegistry" switch, ConfigFile parameter will be used in pass 2. Otherwise ConfigFile parameter will be ignored in pass 2. If no ConfigFile will be entered on the command line, a file selection dialogbox will open to select the config file.
Commit	optional	Add this Parameter to make the Script actually do anything. If missing, the configuration based on the used ConfigFile will be written to output window to get verified before application. Note: commit will be automatically \$true when in pass 2 except when changing the parameterset to "View"
OverwriteConfigInRegistry	optional	Has only effect if running pass 2. ignores registry setting for ConfigFile and setup account and uses that one in the command parameter instead. (this parameter should be used with care as it allows to mix different ConfigFiles)
CleanUp	optional	This will clean up the registry from temporary setup status entries. Can be used either in any step (during step 1 only together with the "commit" switch) to automatically remove temporary setup status entries after successful installation. When calling CleanUp after successful installation, it removes left over registry entries from ADCS installation.
ViewConfig	optional	Allows to any time reviewing the configuration. ViewConfig will not check for any installation prerequisites and cannot be used with any of the other switches above.
Help	optional	Display help

Table 1: Arguments accepted by the Script

List of Figures

Figure 1: Sample CA Configuration files.....	4
Figure 2: Running the Script with -Help Argument.....	5
Figure 3: Result of step 1 run of the script. The text includes the CSR file name and location.	6
Figure 4: Export the issued certificate in DER encoded binary format.....	6
Figure 5: Save the certificate file as "certnew.cer" into the script's root folder.....	6
Figure 6: Script Run without the -Commit Argument, to check for errors.....	7
Figure 7: Script Run without the -Commit Argument, using sample Data with Errors in the XML File	8

List of Tables

Table 3: Arguments accepted by the Script	9
---	---