How to self-sign your own PowerShell scripts

Introduction

Since security has become more and more of a concern around the globe and more and more people have started to develop all kinds of applications and scripts, this tutorial will explain how you can sign scripts you have developed, and also briefly explain what PowerShell execution policies are and how to change it so that only signed scripts and binaries can be run on your machine.

A word about PowerShell execution policies

There are four different execution policy levels, and I will explain them briefly here.

- **Restricted**
  Can not run any scripts. Default execution policy. Can run PS commands interactive only.
- **AllSigned**
  Can run scripts. All scripts and configuration files must be signed by a publisher that you trust. Opens you to the risk of running signed (but possibly malicious) scripts, after confirming that you trust the publisher.
- **RemoteSigned**
  Can run scripts. All scripts and configuration files downloaded from communication applications such as Microsoft Outlook, Internet Explorer and Outlook Express, must be signed by a publisher that you trust. Opens you to the risk of running malicious scripts not downloaded from these applications, without any warning prompts.
- **Unrestricted**
  Can run scripts. All scripts and configuration files downloaded from communication applications run after confirming that you understand the file originated from the Internet. No digital signature is required. Opens you to the risk of running unsigned, possibly malicious scripts.

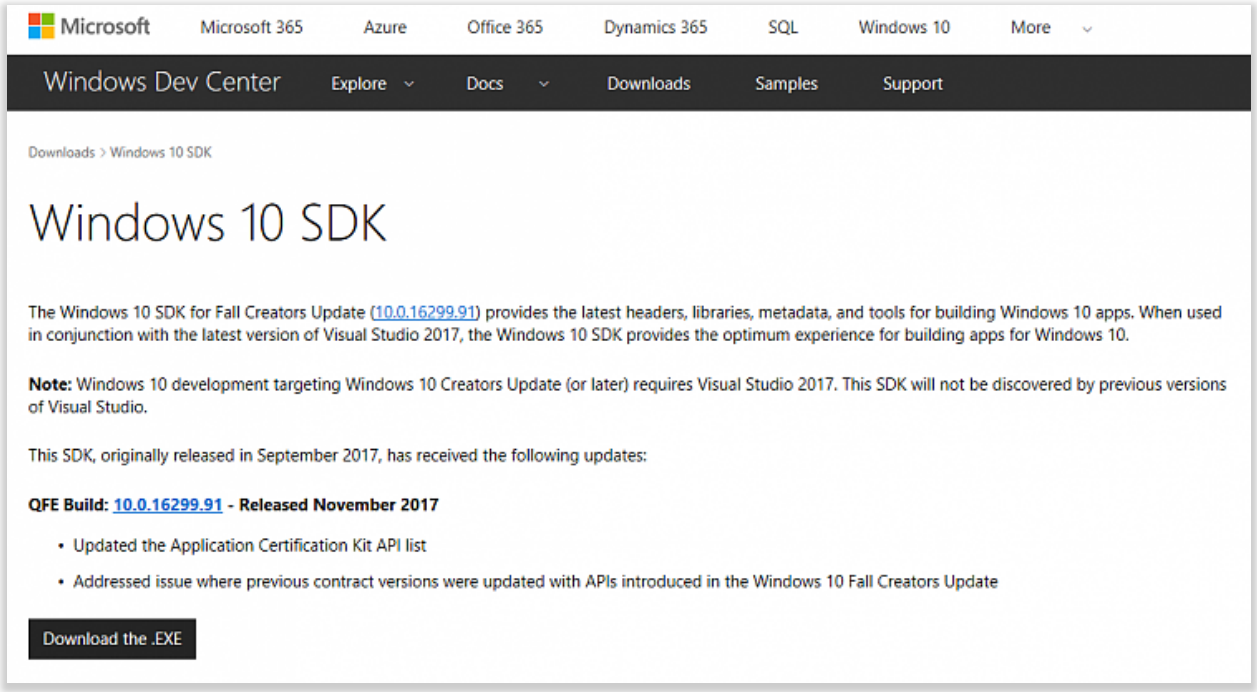
Getting the necessary tools for creating a self-signed

certificate

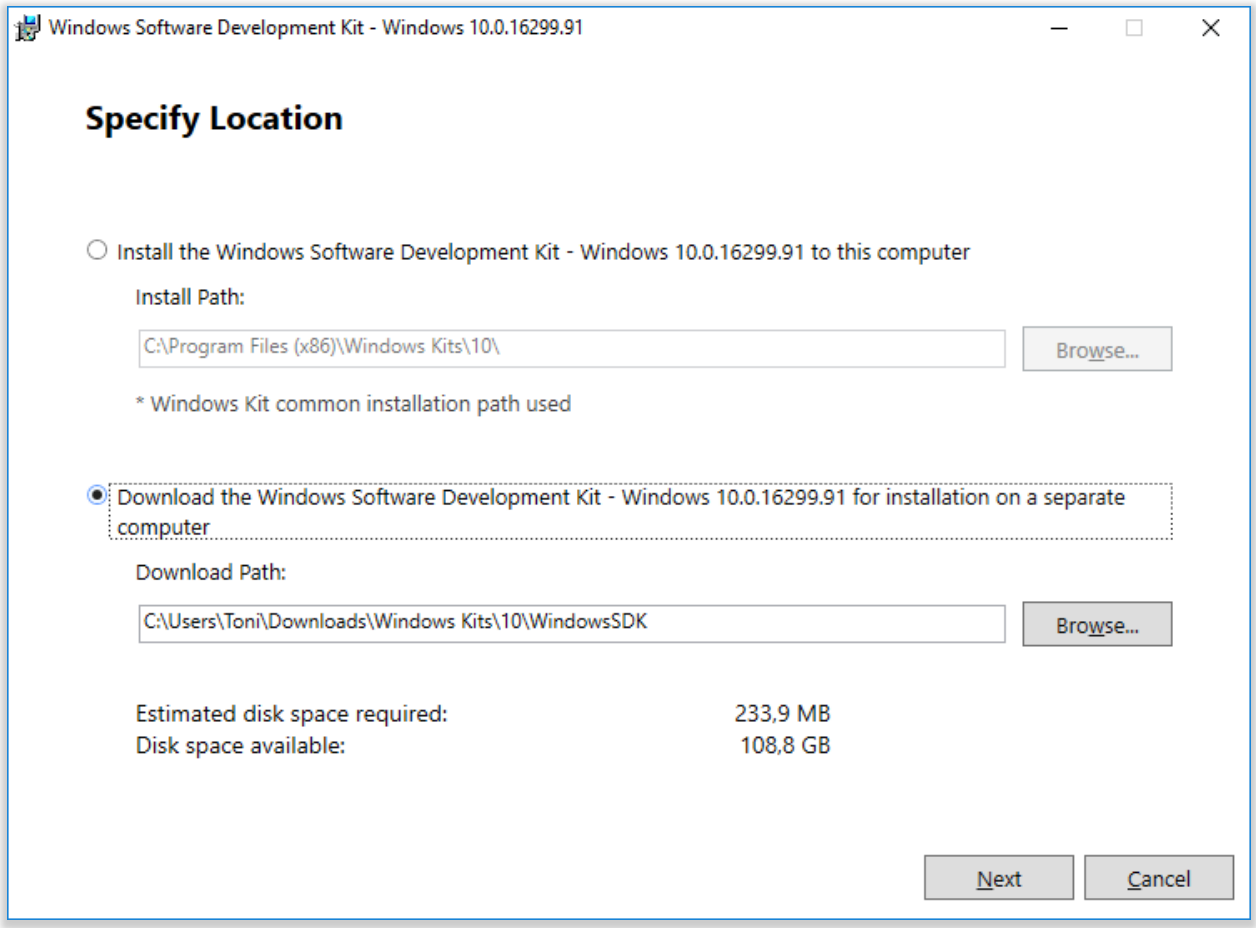
You will either need to have Visual Studio installed or manually download the Windows SDK. In order to save space and bandwidth, this section will explain how to get the latest SDK, download the minimal required files and only install one package from that download.

Steps to take for the download and installation:

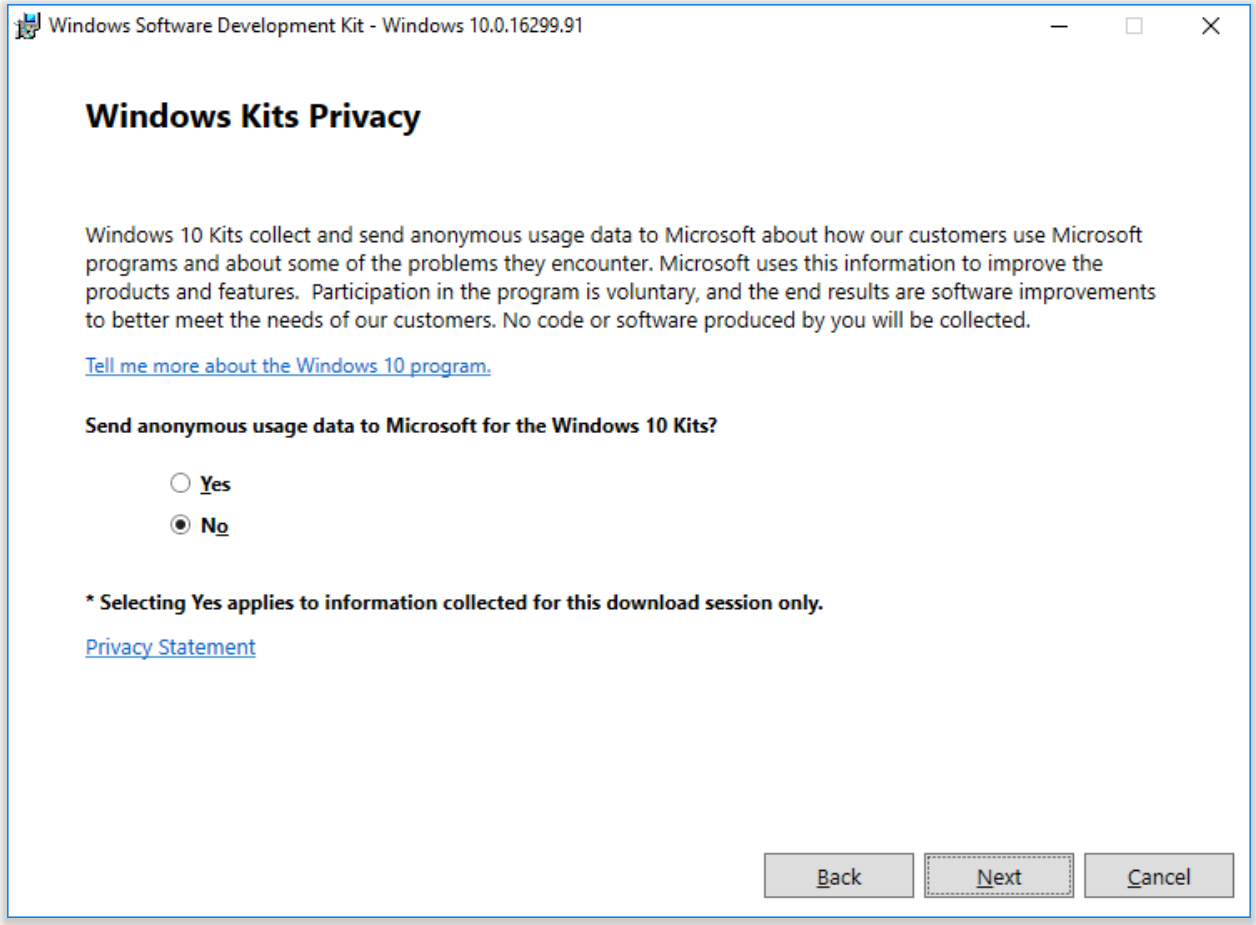
1. Using your web-browser navigate to:
   Download Windows 10 SDK - Latest
   or
   Download Windows 10 SDK - Archive
2. Download the latest SDK installer for Windows (as of writing it is 10.0.16299.91 - Released November 2017)



3. Execute the installer
4. Select Download the Windows Software Development Kit and take note of the path and click next

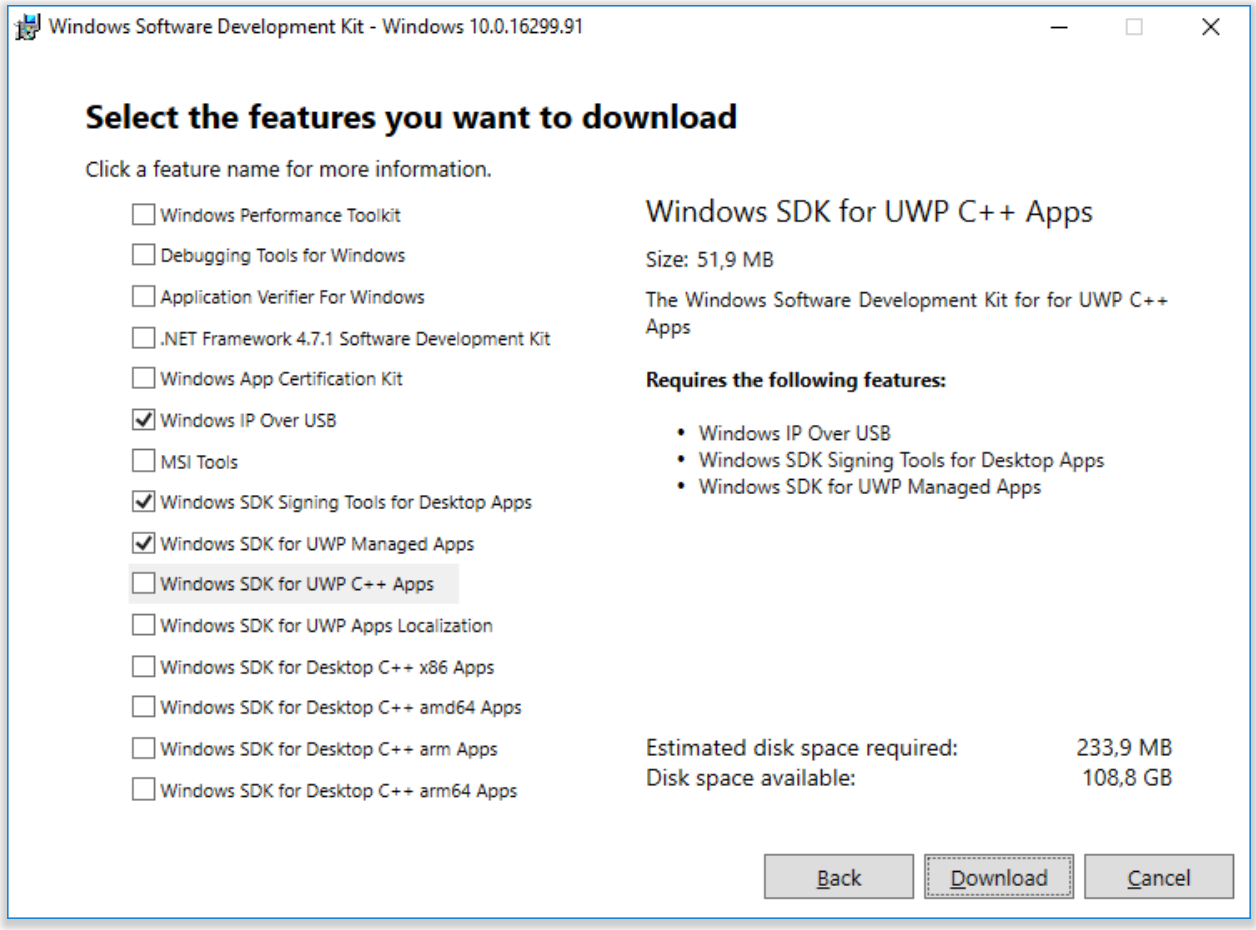


5. On Privacy page, click NO and then next

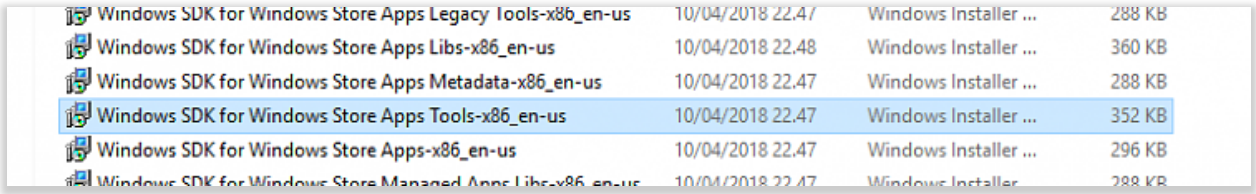


6. On Select the features you want to download page unselect everything, and then only select "Windows SDK for UWP managed apps". 2 other boxes will

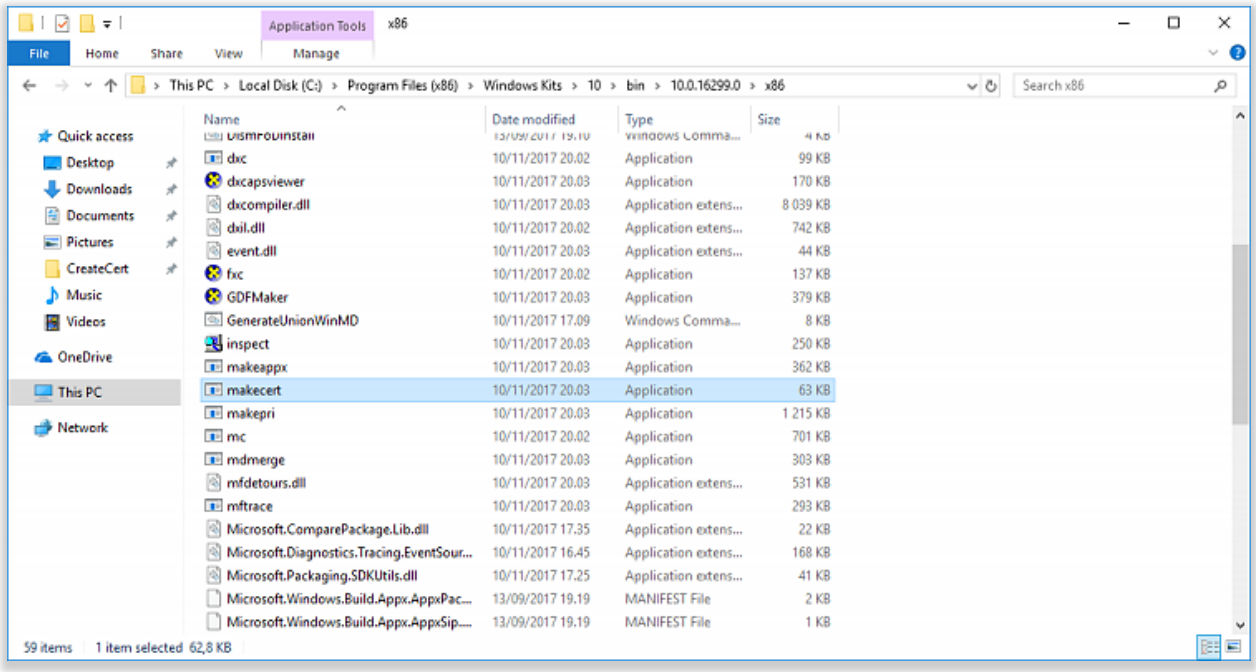
automatically be selected. Click Download.



7. Once the download completes, navigate to the "Windows Kits" folder where you just downloaded the installers.
8. Navigate deeper into the folder structure (\10\WindowsSDK\Installers)
9. Locate the file *"Windows SDK for Windows Store Apps Tools-x86_en-us.msi"* and install it. Note, you will not get any prompts when the installer has completed.



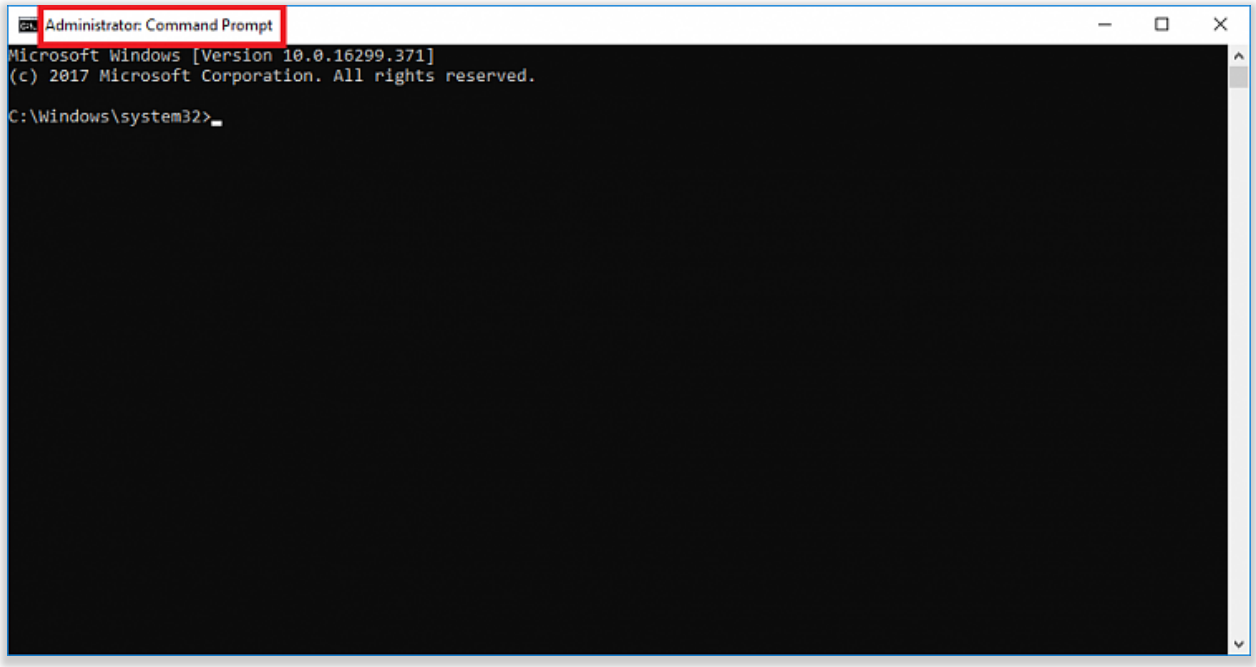
10. Now you should have the required file installed (C:\Program Files (x86)\Windows Kits\10\bin\10.0.16299.0\x86\makecert.exe)



Creating the Certificate

Steps to take:

1. Open up Command Prompt with Elevated privileges.



2. Create a working directory

   Code:

```
md \TEMP
```

3. Navigate to the working directory

   Code:

```
cd \TEMP
```

4. Check path variables and take note of the last character.

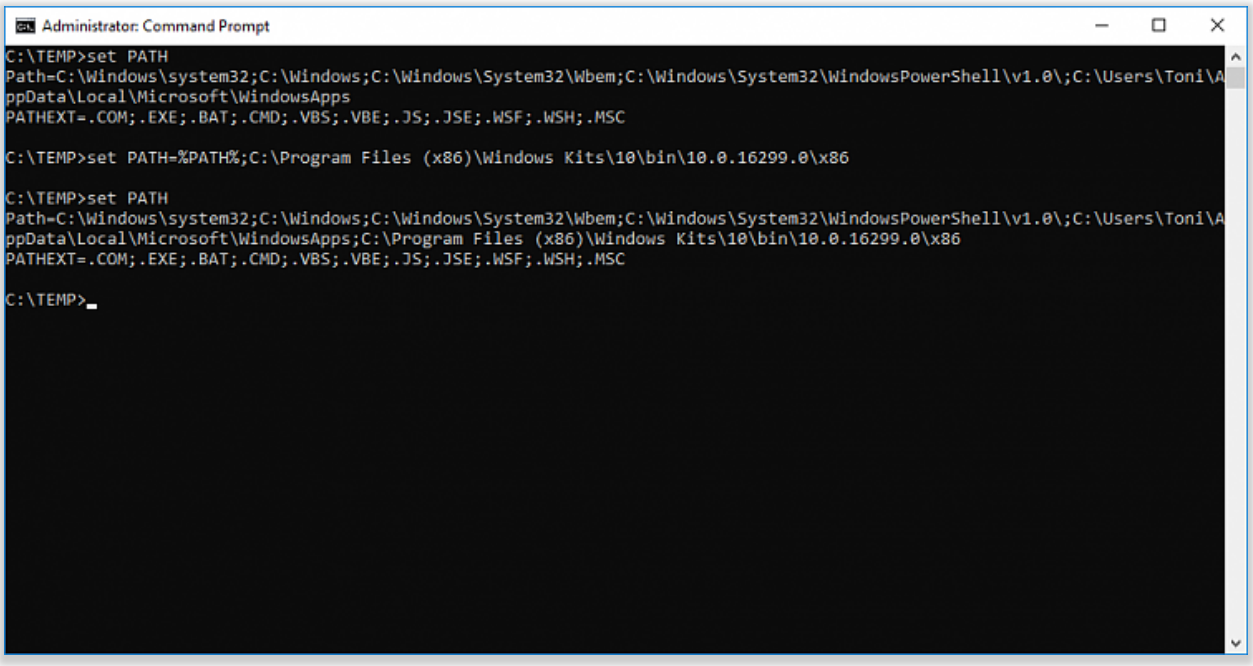
   Code:
   ```
   set PATH
   ```

5. Create a temporary path to the folder where the required certificate tool (makecert.exe) is located.
   If the last character in your path was a semicolon then enter this code:

   Code:
   ```
   set PATH=%PATH%C:\Program Files (x86)\Windows Ki
   ```

   else enter this code:

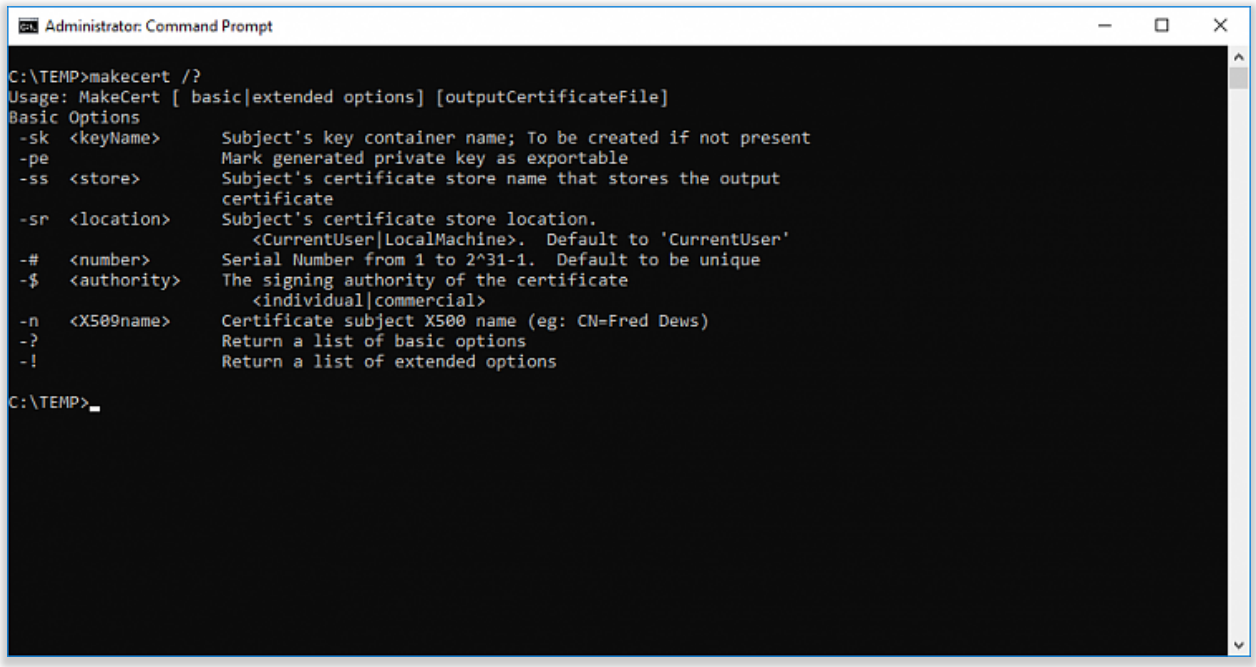
   Code:
   ```
   set PATH=%PATH%;C:\Program Files (x86)\Windows K
   ```



   You can also make the path stick permanently by using setx instead of set.

6. Verify that makecert.exe works

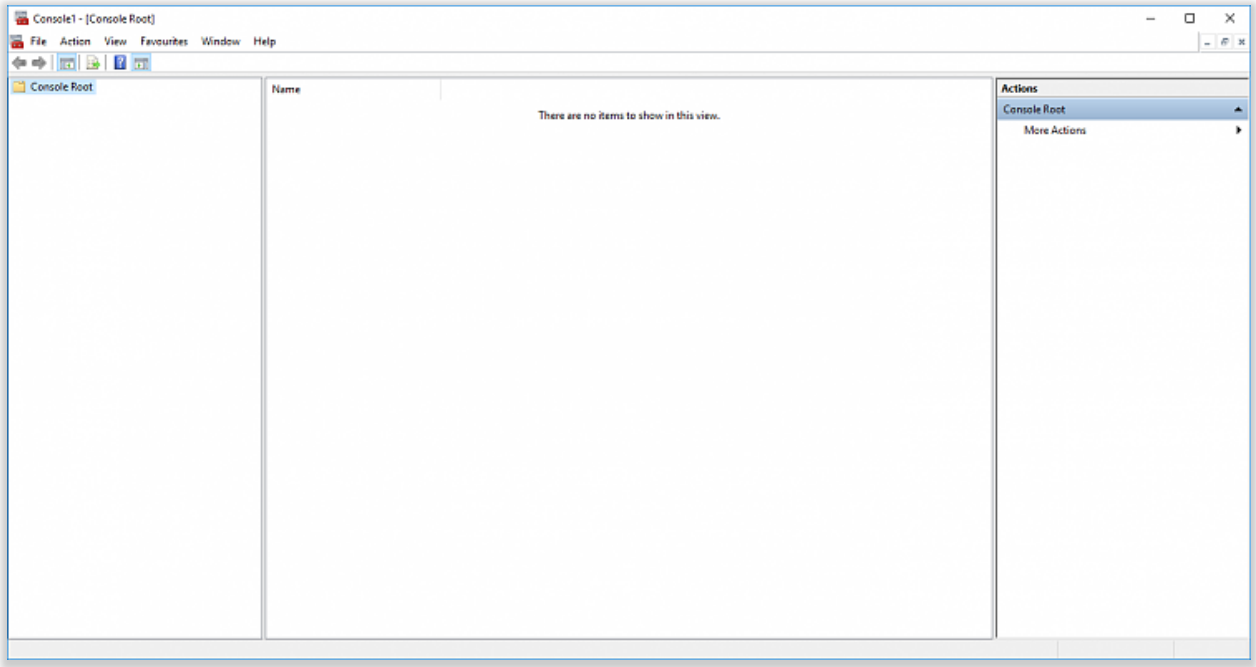
   Code:
   ```
   makecert /?
   ```



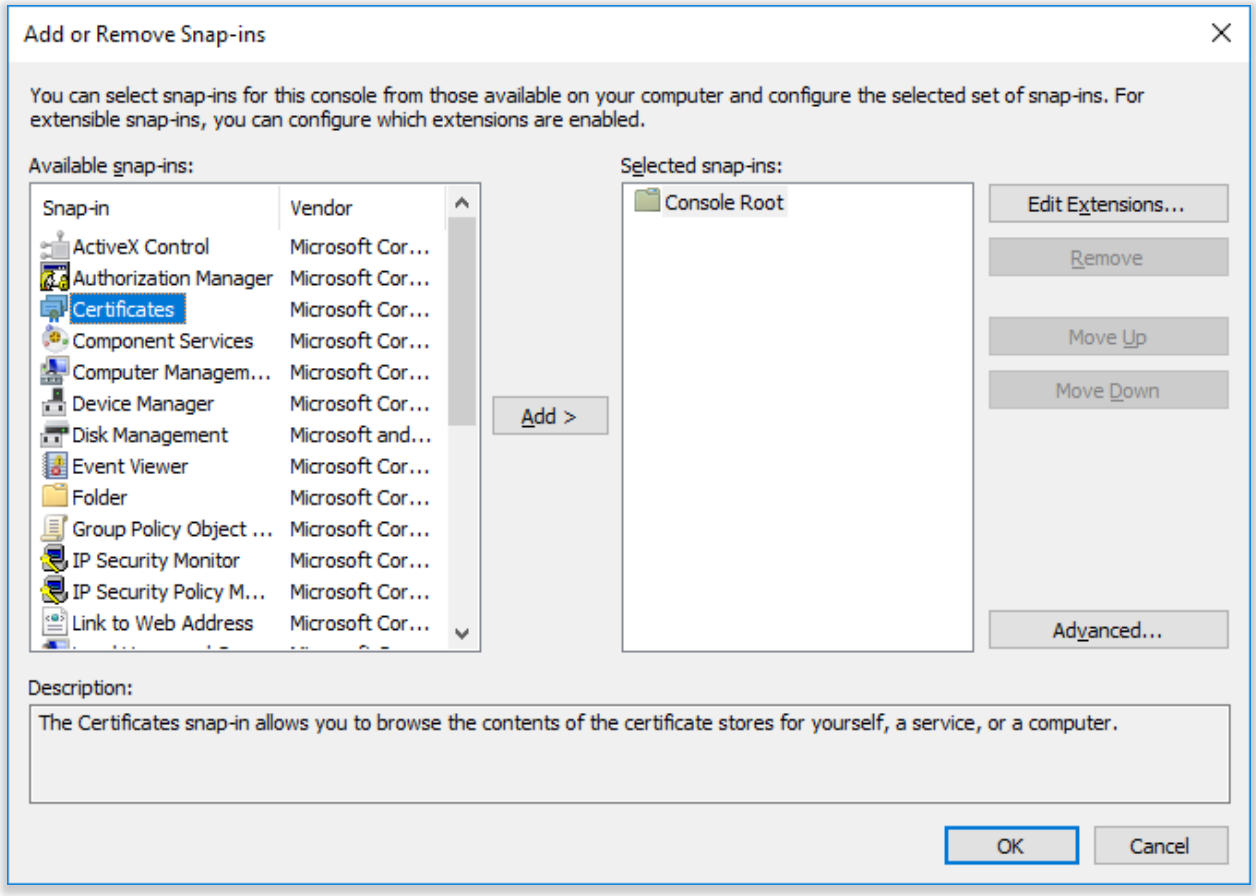
7. Open Microsoft Management Console.

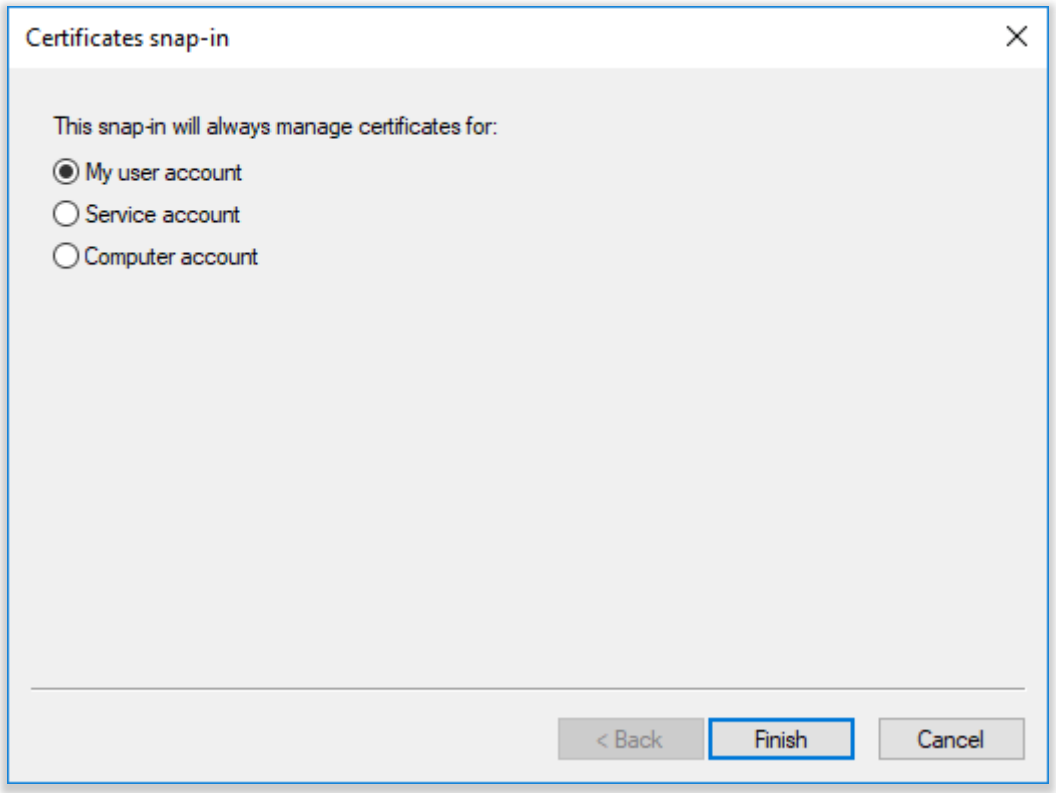
Code:

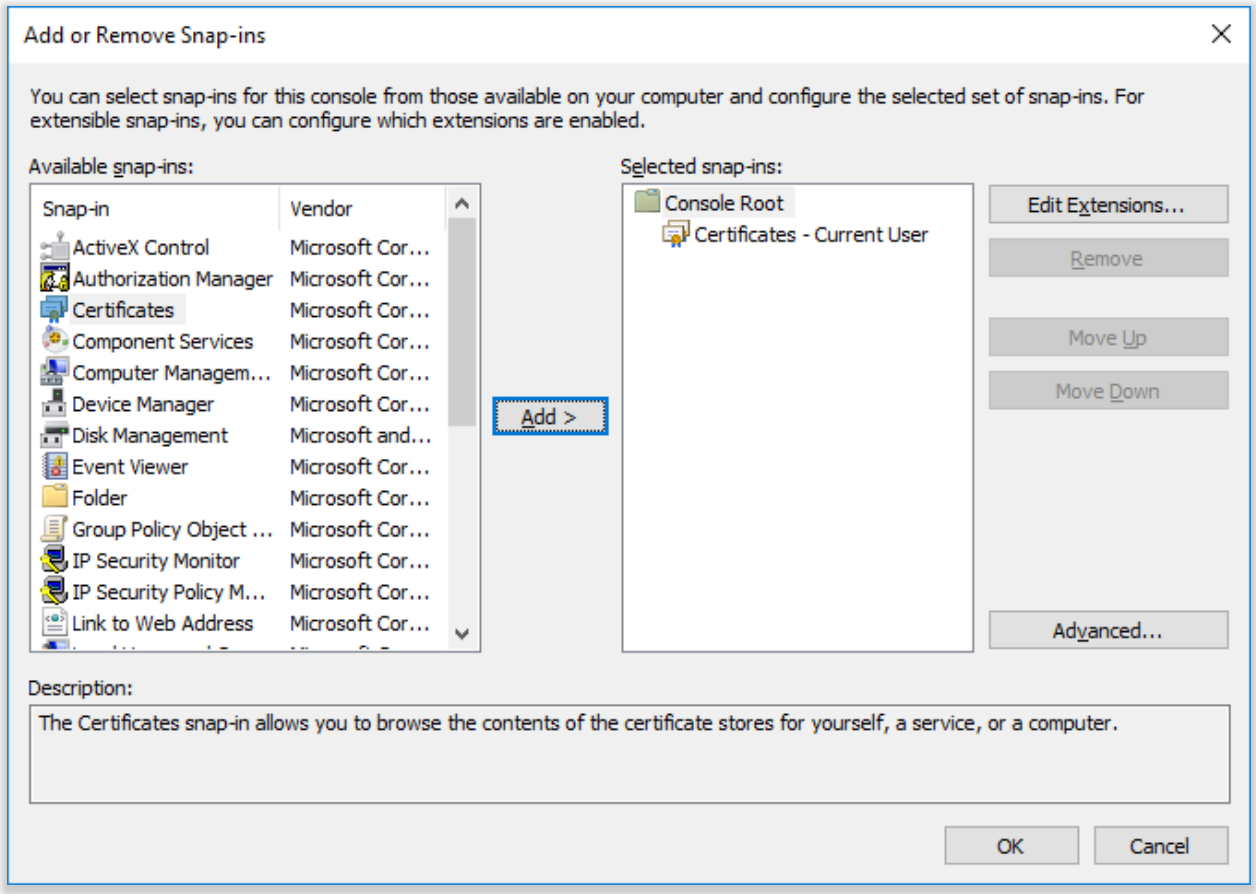
```
mmc.exe
```



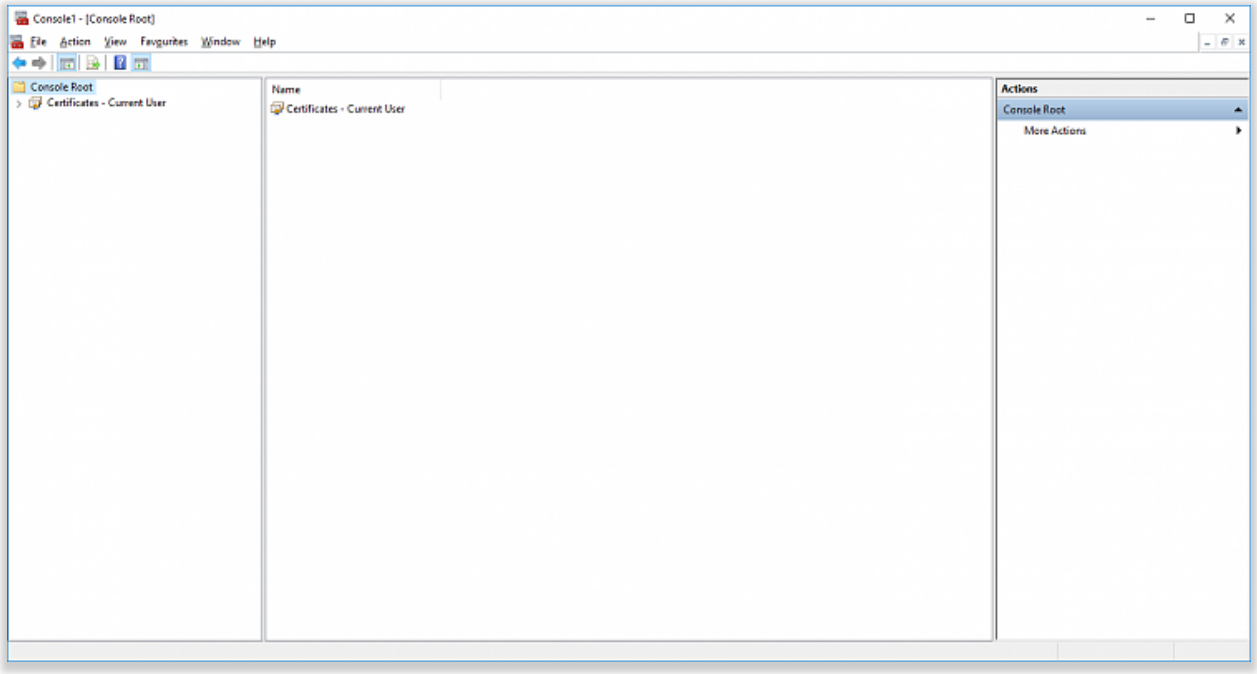
8. Click Add/Remove Certificate Snap-in from File-menu




Certificates snap-in
This snap-in will always manage certificates for:
My user account
Service account
Computer account
< Back
Finish
Cancel


Add or Remove Snap-ins
You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.
Available snap-ins:
Snap-in
Vendor
ActiveX Control Microsoft Cor...
Authorization Manager Microsoft Cor...
Certificates Microsoft Cor...
Component Services Microsoft Cor...
Computer Managem... Microsoft Cor...
Device Manager Microsoft Cor...
Disk Management Microsoft and...
Event Viewer Microsoft Cor...
Folder Microsoft Cor...
Group Policy Object ... Microsoft Cor...
IP Security Monitor Microsoft Cor...
IP Security Policy M... Microsoft Cor...
Link to Web Address Microsoft Cor...
Add >
Selected snap-ins:
Console Root
Certificates - Current User
Edit Extensions...
Remove
Move Up
Move Down
Advanced...
Description:
The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.
OK
Cancel

9. Now your MMC should look like this

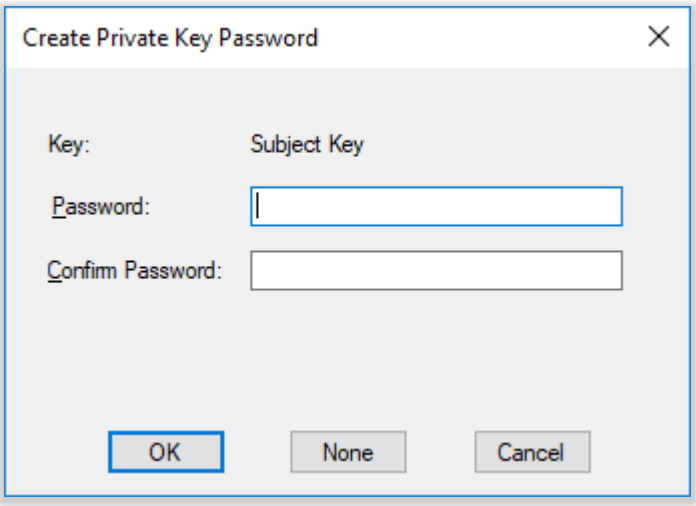


10. Create a Certificate Authority using CMD

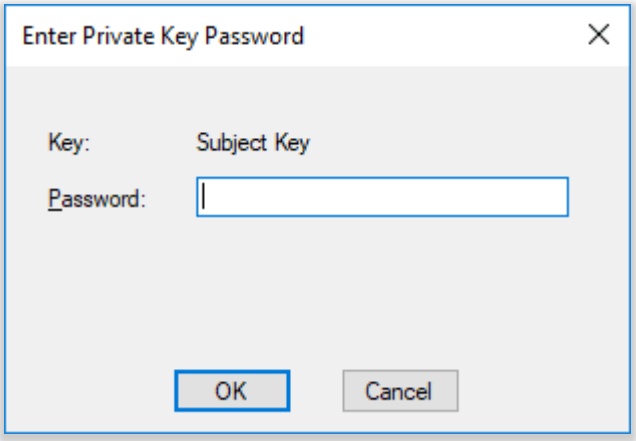
Code:

```
makecert -n "CN=PowerShell Local Certificate Roo
```

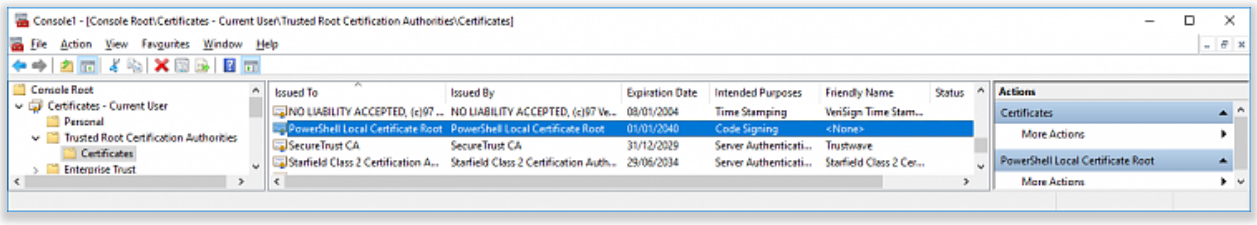
11. Give it a password



12. Verify your password



13. Now your MMC looks like this (may have to refresh by using F5-key)

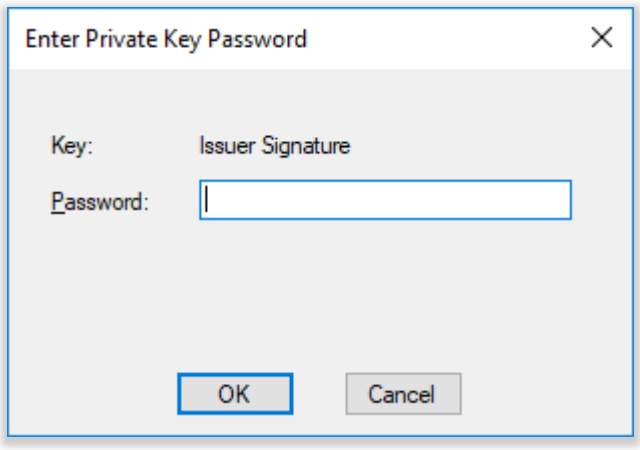


14. Create Personal Certificate

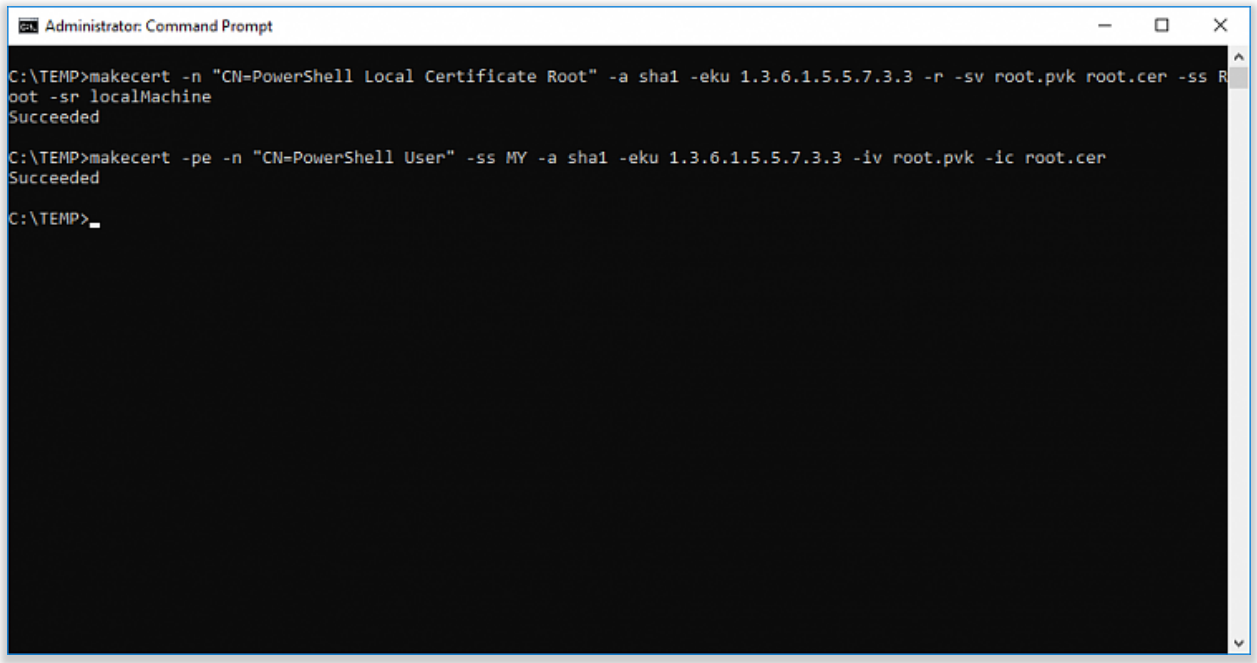
Code:

```
makecert -pe -n "CN=PowerShell User" -ss MY -a s
```

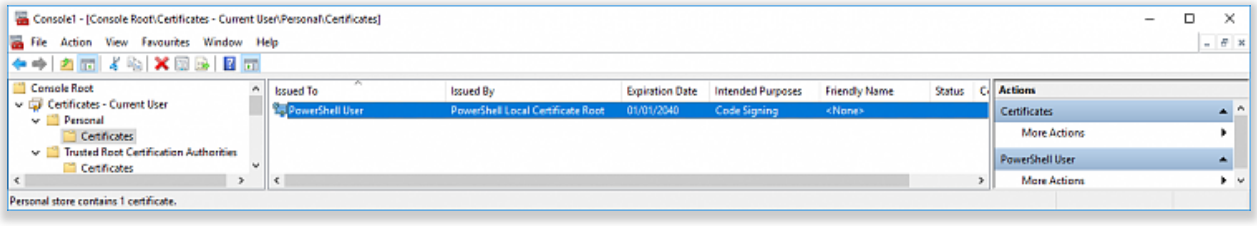
15. Enter your password



16. View of your console after entering the 2 commands



17. Now your MMC looks like this (may have to refresh by using F5-key)



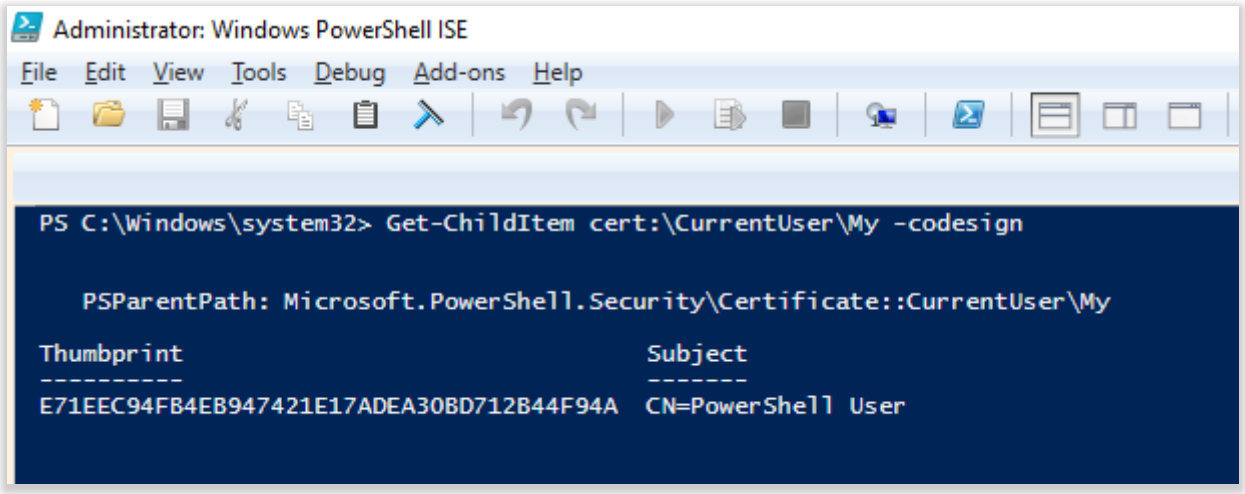
Verifying that we can sign PS scripts

Steps to take:

1. Open elevated PowerShell ISE
2. Checking that we can access our certificate

   Code:

   ```
   Get-ChildItem cert:\CurrentUser\My -codesign
   ```

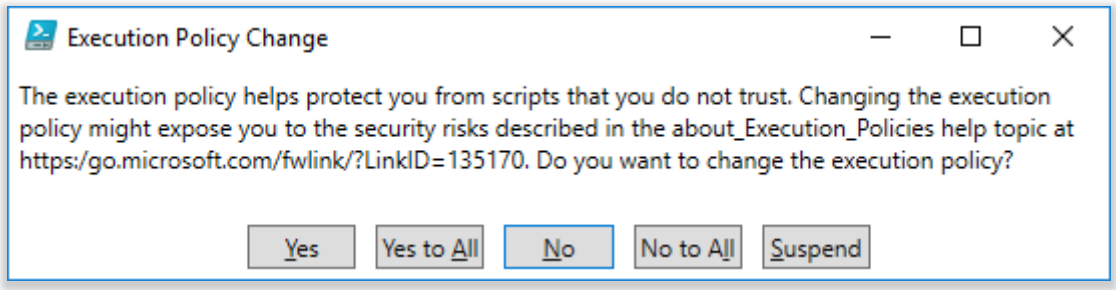
   

3. Changing execution policy to AllSigned

   Code:

   ```
   Set-ExecutionPolicy AllSigned
   ```

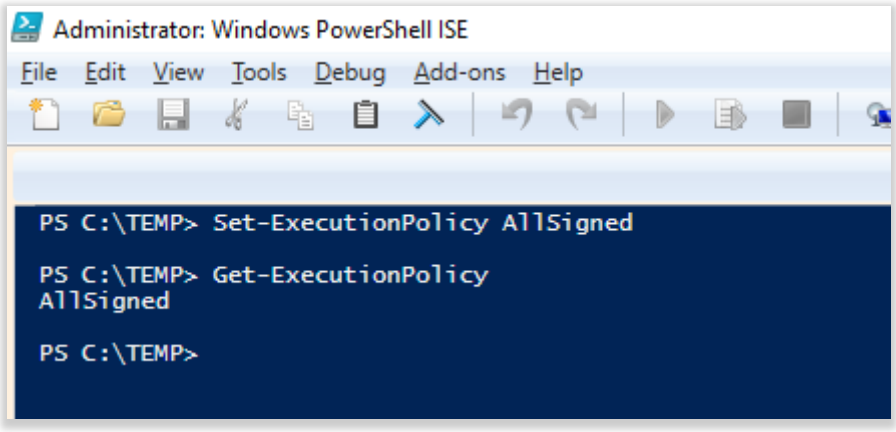
4. Answer Yes at the prompt

5. Verifying that everything got set properly

   Code:

   ```
   Get-ExecutionPolicy
   ```



6. Navigate to our working folder

   Code:

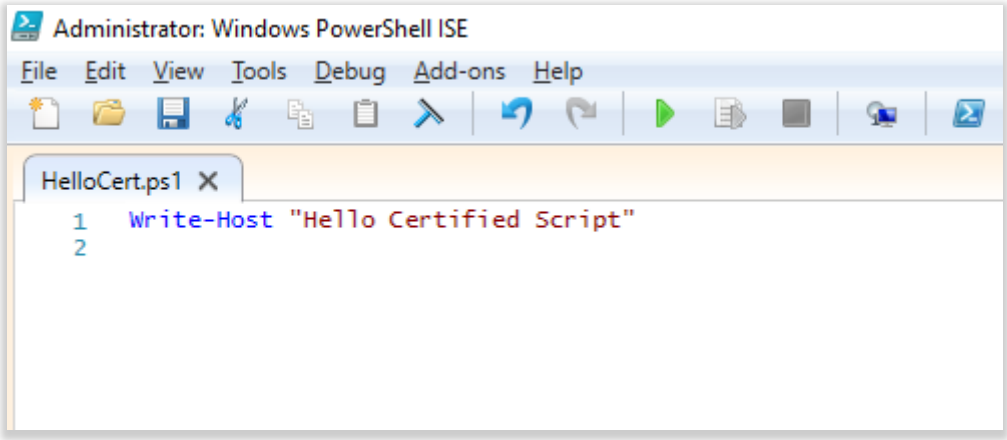
```
cd \TEMP
```

7. Next we create a script that we will test, sign and test again
8. In PSISE select File>New
9. Enter the following code and press enter

   Code:

```
Write-Host "Hello Certified Script"
```

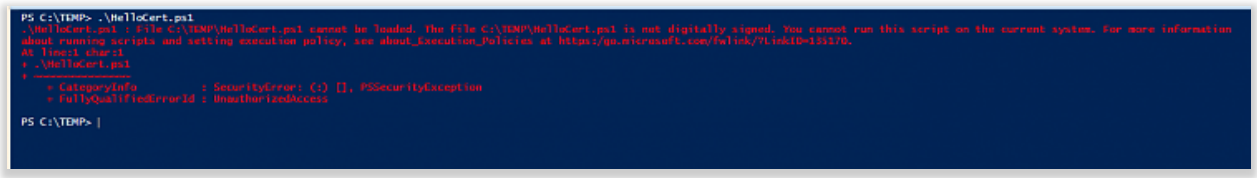
10. Save the script as HelloCert.ps1



11. Test the script (notice you will get an error)

    Code:

```
'\HelloCert.ps1
```



12. Sign the script

    Code:

```
Set-AuthenticodeSignature HelloCert.ps1 @(Get-Ch
```

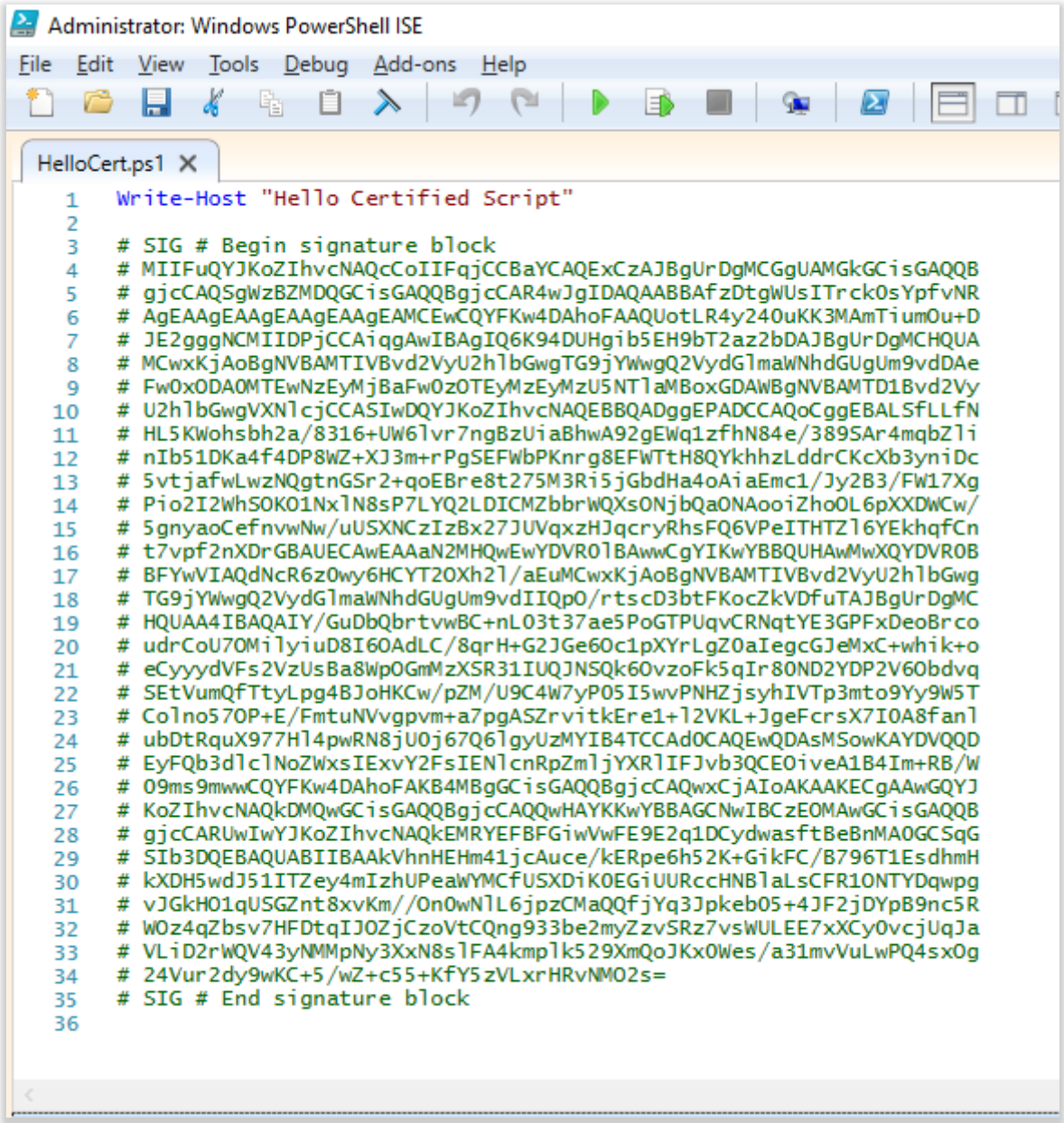
```
PS C:\TEMP> Set-AuthenticodeSignature HelloCert.ps1 @(Get-ChildItem cert:\CurrentUser\My -codesign)[0]

    Directory: C:\TEMP

SignerCertificate                         Status                                 Path
-----------------                         ------                                 ----
E71EEC94FB4EB947421E17ADEA30BD712B44F94A  Valid                                  HelloCert.ps1


PS C:\TEMP>
```

13. Close the current HelloCert.ps1 you were earlier editing and open the file again (note the difference)



14. Run the script

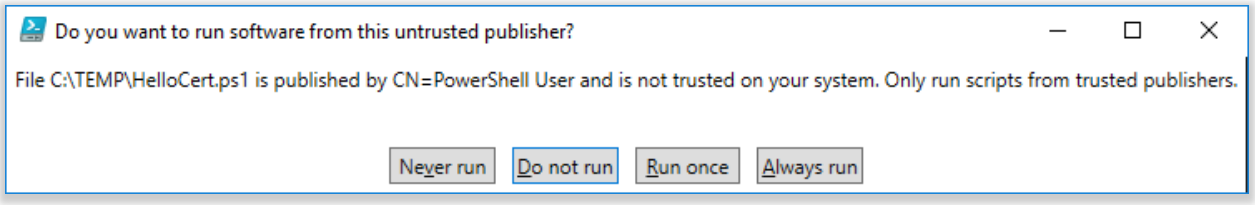
Code:

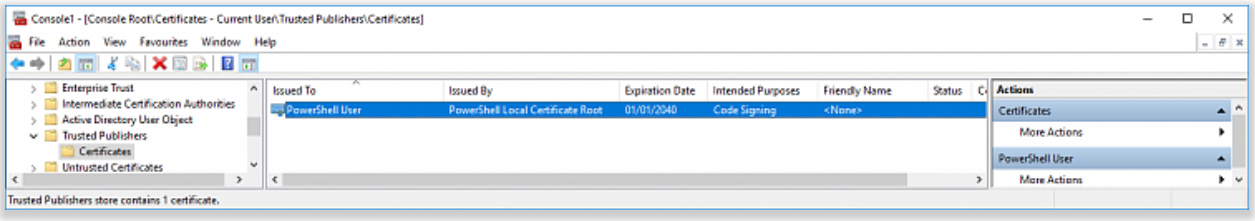
```
.\HelloCert.ps1
```

```
PS C:\TEMP> .\HelloCert.ps1
Hello Certified Script

PS C:\TEMP>
```

15. Press Always run to add the certificate to Trusted Publishers





16. All should work now as expected and you have maximum security enabled that still allows you to run scripts.

CONGRATULATIONS! You are now officially a PS-script signing guru!