INNES appi AZURE AD Powershell module

Introduction

This set of *Powershell* functions allows to:

- create an *Azure Active Directory* application, with the `New-AppiAADApplication` function,
- remove an *Azure Active Directory* application, with the `Remove-AppiAADApplication` function.

These functions are defined in the `PSAppi` PowerShell module stored in the `Modules\PSAppi\` directory.

The result of the *Powershell* function is also stored in a JSON file. Edit the file and store preciously the values which could be required for your Appi:

- the `clientId` value,
- the `tenantId` value,
- the `clientSecret` value.

Security

By default, the execution of local *Powershell* scripts are not allowed. You can change their execution rights by changing the *PowerShell* security policy. This modification has to be done once with the `Set-ExecutionPolicy` *Powershell* function. Your organization may have to change it according to your security rules.

For example, to authorize the execution of all scripts, launch a *Powershell* console with administrator rights, and type:

```
PS > Set-ExecutionPolicy -ExecutionPolicy Unrestricted -scope CurrentUser
```

For further information, look at the cmdlet `Set-ExecutionPolicy` help page.

If you cannot allow the execution of unsigned local scripts, you can install the provided certificate in the list of authorized root certificates with the command:

```
PS > cd <your_path_to_the_scripts>\Powershell_Innes_Appi\Certificate\
PS > Import-PfxCertificate -FilePath InnesCodeSigningRootCA_1.pfx -
CertStoreLocation cert:\CurrentUser\Root -Password $(ConvertTo-SecureString "1234"
-AsPlainText -Force)
```

To import the .pfx certificate, you can also use the MS-Windows application `certmgr.msc`, select the `Trusted Root Certification Authorities`, right clic on `All Tasks`, select the `Import` item, select the file and enter the password `1234`. When ended, close the current Powershell console.

Prerequisite

Install the AzureAD module

Install the *AzureAD* module with the command below:

```
PS > Install-Module -name AzureAD -scope CurrentUser
```

Dependency

If this message is prompted, enter Y.

```
The NuGet supplier is required to continue
PowerShellGet requires the NuGet vendor, version 2.8.5.201 or later, to interact
with the repositories.
The NuGet provider must be available in "C:\Program
Files\PackageManagement\ProviderAssemblies" or "C:\Users\
<username>\AppData\Local\PackageManagement\ProviderAssemblies".
You can also install the provider NuGet by executing the command "Install-
PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force". Do you want that
PowerShellGet installs and imports the NuGet provider now?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"):
```

If this message is prompted, enter Y.

```
Unapproved repository
You install the modules from an unapproved repository. If you approve this
repository, change its InstallationPolicy value by running the Set-PSRepository
command applet. Do you really want to install From PSGallery ?
[Y] Yes [T] Yes for all [N] No [U] No for all [S] Suspend [?] Help (default is
"N"):
```

Usage

To use one of the Innes Appi *Powershell* modules, you have 3 possibilities:

- Either copy the directories under `Modules\` into a standard *Powershell* module installation directory, for example "C:\Program Files\WindowsPowerShell\Modules". Then launch a *Powershell* console.
- Or redefine the search variable for *Powershell* modules (the `$Env:PSModulePath` *Powershell* variable) each time you will use theses functions. In this case, launch a *Powershell* console, and type the line below, adapting it to your path. Each time you will launch a new *Powershell* console, you will have to enter it again.

For example:

```
PS > $Env:PSModulePath="$Env:PSModulePath;
<your_path_to_the_scripts>\Powershell_Innes_Appi\Modules"
```

- Or redefine the search variable for *Powershell* modules in the Windows environment variables. For that, add the path `<your_path_to_the_scripts>\Powershell_Innes_Appi\Modules` to the environment variable `PSModulePath`. Then, launch afterwards a *Powershell* console.

To use the functions or get help, you must then import the module(s) with the `Import-Module` function. Example:

```
PS > Import-Module PSAppi
```

Depending on how your get the scripts, you may have this following warning:

```
Security Warning Run only scripts that you trust. While scripts from the Internet
can be useful, this script can potentially harm your computer. Do you want to run
\server\scripts\my.ps1? [D] Do not run [R] Run once [S] Suspend [?] Help (default
is "D"):
```

To avoid this message, you can unblock the script files (to do only once):

```
PS > cd <your_path_to_the_scripts>\Powershell_Innes_Appi\
PS > dir -Recurse | Unblock-File
```

The `Get-Command` function allows you to list the functions defined in a module. Example:

```
PS > Get-Command -Module PSAppi
```

Answer example:

```
CommandType     Name                           Version    Source
-----------     ----                           -------    ------
Function        New-AppiAADApplication         1.10.10    PSAppi
Function        Remove-AppiAADApplication      1.10.10    PSAppi
```

You can get help on each function of the module by using the standard cmdlet `Get-Help` with options:

- `-detailed`,
- `-full`,
- `-examples`.

Example:

```
PS > Get-Help -detailed New-AppiAADApplication
```

Example to create an Azure Active Directory EWS application

```
PS > $signmeeting = New-AppiAADApplication -appname "SignMeeting"  -authorizations
"ews"
```

Warning: clicking on a Powershell window can suspend the command. In this case click again in the window to resume the command.

A login popup is displayed . Enter once your EWS credentials. This message is then displayed in a *Powershell* context.

```
You must log into an administrator account of your organization and grant the
necessary permissions.
A consent request will be sent within 30 seconds in your browser.
```

After 30 seconds, a login popup should be prompted (https://login.microsoftonline.com/) automatically in your default Web browser. Enter again your EWS credentials. A new popup message with the *Permission requested, review for your organization* title is prompted in your Web browser. Press on the `Accept` button. Then a message is displayed in your Web browser showing that the consent is successful: *Success of the consent request.*

You can view the data of the created application by typing the following command :

```
PS > $signmeeting
Name                           Value
----                           -----
clientId                       xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
objectId                       xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
spId                           xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
name                           SignMeeting
tenantId                       xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
clientSecret                   xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

The result of the *Powershell* function is also stored in a JSON file: `SignMeeting.json`. Edit the file and store preciously the values required for your Appi:

- the `clientId` value,
- the `tenantId` value,
- the `clientSecret` value.

Example to delete an Azure Active Directory application

```
PS > Remove-AppiAADApplication -appname "SignMeeting"
```

A login popup is opened. Enter your EWS credentials.