Innes Powershell Modules - User manual

Introduction

This set of *Powershell* functions supports several features:

- Retrieve informations about your `Plugncast` server: licenses, domains and targets, appis installed, content models installed.
- Create for example a custom `Plugncast` URI file from an Excel file with the function `New-PncUri`.

These functions are defined into a *Powershell* module named `PSPnc` which is located at the `Modules/PSPnc` directory.

Security

By default, local `Powershell` scripts cannot be executed. You can change this behavior by changing the *Powershell* security policy. This modification is done with the `Set-ExecutionPolicy` *Powershell* function, and has to be done only once. Your organization may have to change it according to your security rules.

For example, if you want 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 unsigned local scripts, you must place the certificate provided in the list of authorized root certificates. You can do this with the command:

```
PS > cd <your_path_to_the_scripts>\Powershell_Innes\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`/`Import`, select the file and enter the password `1234`. When ended, close the current *Powershell* console.

Usage

To use an `Innes powershell module`, you have 3 possibilities:

- 1 - Copy the directories under `Modules\` into a standard *Powershell* module installation directory, for example "C:\Program Files\WindowsPowerShell\Modules". Then, launch a *Powershell* console.
- 2 - Redefine the search variable for *Powershell* modules (the `$Env:PSModulePath` Powershell variable) each time you will use theses functions. To do that, launch a *Powershell* console, and type the line above, adapting it to your path. Each time you will launch a new *Powershell* console, you will have to type it again.
  Example:

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

- 3 - Redefine the search variable for *Powershell* modules in the MS-Windows environment variables. For that, add the path `<your_path_to_the_scripts>\Powershell_Innes\Modules` to the environment variable `PSMODULEPATH`. Then, launch a *Powershell* console (don't launch the console before modifying the environment variable).

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

```
PS > Import-Module PSPnc
```

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 prevent from having it, you can unblock the script files (to do only once):

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

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

```
PS > Get-Command -Module PSPnc
```

You can get help on each function of the module by using the standard cmdlet `Get-Help` with option `-full` or `-examples` or `-detailed`.
Example:

```
PS > Get-Help -detailed Get-PncContentModelInstalled
```

Examples

In the directory `Examples`, you can find different *Powershell* scripts which uses the functions of the modules.
Before using it, refer to the previous paragraph to set the environment correctly, and launch a *Powershell* console.
As all the examples invoke the necessary `Import-Module` function, you don't have to call it if you use only the examples.
You can get help on each example scripts, for example:

```
PS > Get-Help -detailed .\Examples\Example1\Get-PncInformation.ps1
```

Example 1

The script `Examples\Example1\Get-PncInformation` is an example to retrieve informations about the `Plugncast Server`. It uses the module `PSPnc`.
Example:

```
PS > cd <your_path_to_the_scripts>\Powershell_Innes\Examples\Example1\
PS > .\Get-PncInformation.ps1 -UrlHost 192.168.1.15 -UrlLogin superadmin -
UrlPassword superadmin -UrlPort 443 -LogFile result.txt
```

If any error occurs, look at the logfile (`result.txt` in the example) to see what the problem may be.

Example 2

The script `Examples\Example2\New-PncUriFromExcel` is an example to create URI files from data stored in a MS-Excel file and associated thumbnails in png files. It uses the module `PSPnc`.
It requires the installation of the `PSExcel` *Powershell* module, if not already installed before. You have to launch an other *Powershell* console, with administrator rights, and then, install the module:

```
PS > Install-Module -Name PSExcel
```

Then, quit this console, and go back to the *Powershell* console with the proper environment set. Example:

```
PS > cd <your_path_to_the_scripts>\Powershell_Innes\Examples\Example2\
PS > .\New-PncUriFromExcel.ps1 -excelFileList .\channels.xlsx -outputDirectory
.\output\
```