PowerShell for ProjectWise

How does it work?

The number of ProjectWise PowerShell commands are constantly growing.
Through ProjectWise for PowerShell it is possible to run commands in ProjectWise.

How does PowerShell work, and how can I benefit from the fast growing ProjectWise commands.
First of all, how is PowerShell for ProjectWise activated.
We will look at some simple commands and show some more active scripting.

Warning: When logged in as Administrator with -Admin anything can be done through PowerShell.
'Watch out for long Query Sessions'

Christian J. Jakobsen

COWI

Some useful links.

- Microsoft
  - All information about PowerShell. Getting Started
- Bentley ProjectWise
  - Bentley Community.
- PowerWise
  - This is a Bentley site.
- POSHGUI
  - **PO**wer**SH**ell**GUI**
- GitHub
- PS2EXE-GUI

Short presentation of the PowerShell Syntax.

```
Import-Module -Name pwps_dab –Verbose
Open-PWConnection -GUI -Admin
$EName = 'Complex'
$Env = get-PWEnvironmentColumns -EnvironmentName $EName
$Env = $Env | where {$_.Name -Like 'Doc*'}
foreach ($n in $Env) {
        Write-Host $n.Name ',' $n.Size ',' $n.Precision ',' $n.SQLType
       #  Update-PWEnvironmentColumnWidth -EnvironmentName $EName  -ColumnName $n.Name -NewWidth 4
}
Get-PWSession
Close-PWConnection
New-PWLogin
```

- Get-PWSessions
- Set-PWSession

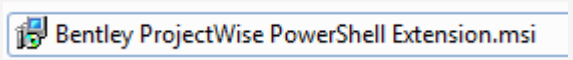
1 for char
10 for varchar
12 for nchar
13 for nvarchar

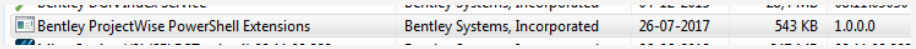
pw.cowi.com:DKLYPW5Test5

COWI

Installing Bentley ProjectWise PowerShell.

- Two different modules exist.
  - PWPS
    - Installation from an MSI-files.
    - Can be found in the ProjectWise Administrator package.
      - The Administrator module must be installed before the PW PowerShell?
        - C:\Program Files (x86)\Bentley\ProjectWise\bin\PowerShell





  - PWPS_DAB
    - Installation through PowerShell.
    - Use 'Install-Module –Name pwps_dab –Verbose'.
    - We will get back to this…

Execution Policy.

- Set-ExecutionPolicy -Scope CurrentUser –ExecutionPolicy Unristricted
  - By default, **PowerShell's execution policy** is set to Restricted; this means that scripts will not run.
  - –ExecutionPolicy
    - **Restricted:** No scripts can be run. Windows PowerShell can be used only in interactive mode.
    - **AllSigned:** Only scripts signed by a trusted publisher can be run.
    - **RemoteSigned:** Downloaded scripts must be signed by a trusted publisher before they can be run.
    - **Unrestricted:** No restrictions; all scripts can be run.
  - –Scope
    - **Process**: The execution policy affects only the current PowerShell process.
    - **CurrentUser:** The execution policy affects only the current user.
    - **LocalMachine:** The execution policy affects all users of the computer. PowerShell as Administrator

  Run 'Windows PowerShell ISE (x86)' as Administrator.

COWI

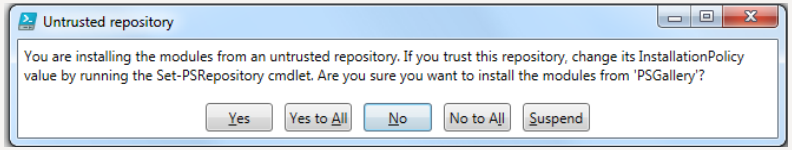
Installing Modules.

- Install-Module –Name xxxx –Scope CurrentUser -Force
  - Install-Module is a command that downloads and installs modules from PSGallery.
  - -Scope AllUsers (Default)
    - %systemdrive%:\Program Files\WindowsPowerShell\Modules
  - -Scope CurrentUser
    - $home\Documents\WindowsPowerShell\Modules
- Import-Module –Name xxxxx -Verbose
  - Import-Module is the vanilla cmdlet to import and load a module that's already installed.
    - When installed, it should automatically Import.
- Get-PSRepository
- Set-PSRepository –Name 'PSGallery' –InstallationPolicy Trusted

COWI

Installing PowerShell for ProjectWise.

- Install PWPS with 'Bentley ProjectWise PowerShell Extension.msi'.
  - ProjectWise Administrator must be installed.
- Install PWPS_DAB module from PowerShell.
  - Run PowerShell as Administrator.
    - Can only install modules as Administrator, if installed in 'Program Files (x86)' folder.
  - Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Unrestricted
    - Now all is unrestricted.
  - Install-Module –Name pwps_dab –Verbose *-Force*
    - This will install the module on client PC.
  - Import-Module -Name pwps_dab –Verbose
    - This will load the module in the session.



COWI

Some useful PowerShell commands.

- Get-PSRepository

```
Name                PackageManagementProvider  InstallationPolicy  SourceLocation
----                -------------------------  ------------------  --------------
PSGallery           NuGet                      Untrusted           https://www.powershellgallery.com/api/v2
```

- Find-Module –Name pwps_dab

```
Version   Name                Type       Repository      Description
-------   ----                ----       ----------      -----------
1.7.4.0   pwps_dab            Module     PSGallery       ProjectWise PowerShell Cmdlets from System Architecture Group
```

- Get-Module –Name pwps_dab or -All
- Get-Command –Module pwps_dab
- Find-Module -Name " ImportExcel"

```
Version   Name                Type       Repository      Description
-------   ----                ----       ----------      -----------
5.3.4     ImportExcel         Module     PSGallery       PowerShellmodule to import/export Excel spreadsheets, without Excel....
```

- $PSVersionTable.PSVersion (Newest version 6.0)
- $env:PSModulePath (Environment PSModulePath)

COWI

Logging in to a DataSource.

- Open-PWConnection (pwps)
  - -DataSourceName 'ProjectwiseLyngby5Test'
  - -SSO –Admin -Gui
- New-PWLogin (pwps_dab)
  - -DataSourceName –Password -UserName
  - -SSO –UseGui -NonAdminLogin
  - -Password is not accepted as clear text.
- Close-PWConnection # Logout

COWI

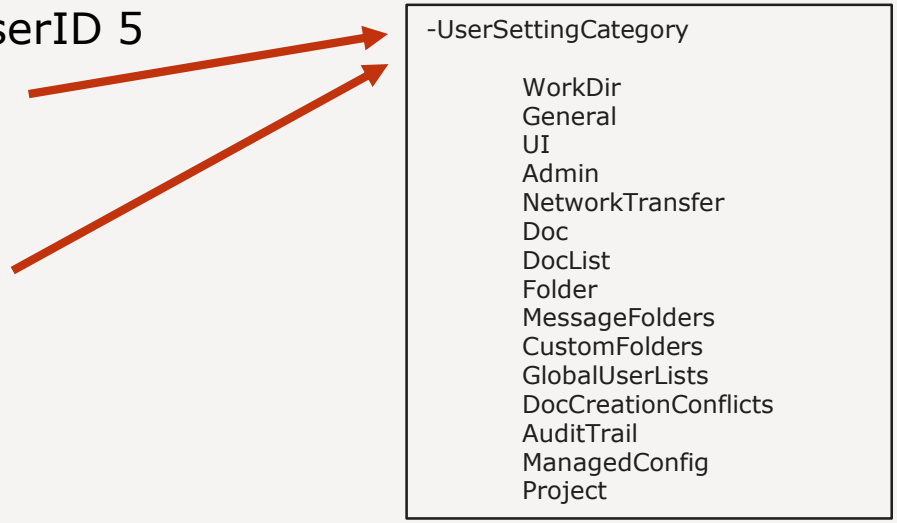
Passwords & Security.

- Save-SecureStringToEncryptedFile
  - Will save a password to a file.
    - Save-SecureStringToEncryptedFile -FileName "C:\Temp\xxx.pas" –Prompt 'Enter Password'
- Get-SecureStringFromEncryptedFile –FileName 'C:\Temp\xxx.pas'
  - Will read the password.
    - $SP = Get-SecureStringFromEncryptedFile -FileName "C:\Temp\xxx.pas"
  - $LoginStatus = New-PWLogin -DatasourceName 'Projectwise Lyngby5 Test -UserName pwadmin -Password $SP
  - $AdPw = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList 'pwadm', $SP
-

COWI

ProjectWise Users.

- Get-PWUser –UserName 'CJJ'
- Set-PWUserDisabled –UserID 5
- New-PWUser –DomainUser -UserName CJJ -Password 'xx' -Emailcjj@cowi.com
- Get-PWUserSetting –UserID 5
  - -UserSettingCategory
  - -UserSettingName
- Set-PWUserSetting
  - -UserSettingCategory
  - -UserSettingName
  - -UserSettingValue



COWI

ProjectWise Selecting Users.

- Get-PWUserByLastLogin -DaysAgo 365 | where {$_.Disabled -eq $False -and $_.type -eq 'Logical' -and (Get-Date $_.LastLogin) -gt (Get-Date '01-01-1979') }| select Email, UserID, UserName, LastLogin, Type, SecProvider | Export-Csv C:\temp\users.csv

- Get-PWUserByLastLogin -DaysAgo 30
  - where {$_.Disabled -eq $False -and $_.Type -eq 'Logical' -and (Get-Date $_.LastLogin) -gt (Get-Date '01-01-1979') }
    - where {$_.Disabled -eq $False -and $_.Type -eq 'Windows' -and (Get-Date $_.LastLogin) -gt (Get-Date '01-01-1979') }
  - Select Email, UserID, UserName, LastLogin, Type, SecProvider
  - Export-Csv C:\temp\users.csv

COWI

Live Demo.

- Five examples.
  - ImportUsers
  - ENVUpdate
  - DisableExtUsers
  - UserLoginTrack
  - CheckProjectsforArchive

COWI

Select Query.

- Select-PWSQL
  - This is the SQL for either just looking at data or writing to Excel.
    - Examples could be…

      - Select-PWSQL -SQLSelectStatement "select * from dms_user"

      - Select-PWSQL -SQLSelectStatement "select * from dms_user" -OutputFile c:\temp\Users.xlsx
      - $Users = Import-excel -Path c:\temp\Users.xlsx # As Array

      - Select-PWSQL -SQLSelectStatement "select o_itemname, o_filename from dms_doc where o_itemname like 'DGN%'"

COWI

Select Query with tables.

- Select-PWSQLDataTable
  - This is the SQL adding data to a table.
    - Examples could be…

      - $Users = Select-PWSQLDataTable -SQLSelectStatement "select * from dms_user"

      - $Users | Format-Table
      - $Users.Table[3]

      - $DGN_Docs = Select-PWSQLDataTable -SQLSelectStatement "select o_itemname, o_filename from dms_doc, o_docguid where o_itemname like 'DGN%'"

COWI

DataTables

- $Users | Format-Table
- $Users.Rows
- $Users.Table[3]
- $Users.Columns.Add("HostName")
- $Users.table[1].Item(4)
- foreach ($n in $Users) {
  - write-host $n.o_username
- }

```
AllowDBNull        : True
AutoIncrement      : False
AutoIncrementSeed  : 0
AutoIncrementStep  : 1
Caption            : o_createdate
ColumnName         : o_createdate
Prefix             :
DataType           : System.DateTime
DateTimeMode       : UnspecifiedLocal
DefaultValue       :
Expression         :
ExtendedProperties : {}
MaxLength          : -1
Namespace          :
Ordinal            : 8
ReadOnly           : False
Table              : {pwadmin, pwprocess, astu_a, astu...}
Unique             : False
ColumnMapping      : Element
Site               :
Container          :
```


COWI

That's it.

Any questions ?