# Powershell

einfach mal eine Sammlung von Codebespielen

- Quick And Dirty
- NT-Accounts
- Choice1
- Choice 2
- InputSample
- Folder
- WARNUNG: Es kann kein Download von URI

# Quick And Dirty

```
function OutputError([string]$errorMessage)
{
Write-Host $errorMessage -ForegroundColor Red
}

$err = Invoke-Expression "./install/sqlCEIPdisable.ps1"
if (!$err) {
# Not True, last operation failed
OutputError "Disable CEIP-Services not successfull."
return
}
Write-Host "Disable CEIP-Services successfull." -ForegroundColor Green
```

---

# NT-Accounts

```
Write-Host "Create NT-Accounts"

$admin = @("ACCOUNTNAME")
$serviceAccounts = @("SERVICENAME")

$PlainPassword = @("PASWORD")
$SecurePassword = $PlainPassword | ConvertTo-SecureString -AsPlainText -Force

#DE 1031
#EN 1033
$CultureID = (Get-Culture).LCID

Write-Host $CultureID

if ($CultureID -eq 1033)
{
Write-Host EN
#Admin-Accounts
for ($i=0; $i -lt $admin.length; $i++){
$rawUserName = $admin[$i]
#Benutzer erstellen
Write-Host $rawUserName
New-LocalUser -Name $rawUserName -Password $SecurePassword -FullName $rawUserName -
Description "AdminAccount"
Set-LocalUser -Name $rawUserName -PasswordNeverExpires 1
#Gruppe setzen
Add-LocalGroupMember -Member $rawUserName -Name Users
Add-LocalGroupMember -Member $rawUserName -Name Administrators
}

#ServiceAccounts
for ($i=0; $i -lt $serviceAccounts.length; $i++){
$rawUserName = $serviceAccounts[$i]
#Benutzer erstellen
Write-Host $rawUserName
New-LocalUser -Name $rawUserName -Password $SecurePassword -FullName $rawUserName -
Description "ServiceAccount"
Set-LocalUser -Name $rawUserName -PasswordNeverExpires 1
#Gruppe setzen
Add-LocalGroupMember -Member $rawUserName -Name Users
Add-LocalGroupMember -Member $rawUserName -Name Administrators
```

```
}
}
else
{
Write-Host DE
#Admin-Accounts
for ($i=0; $i -lt $admin.length; $i++){
$rawUserName = $admin[$i]
#Benutzer erstellen
Write-Host $rawUserName
New-LocalUser -Name $rawUserName -Password $SecurePassword -FullName $rawUserName -
Description "AdminAccount"
Set-LocalUser -Name $rawUserName -PasswordNeverExpires 1
#Gruppe setzen
Add-LocalGroupMember -Member $rawUserName -Name Benutzer
Add-LocalGroupMember -Member $rawUserName -Name Administratoren
}

#ServiceAccounts
for ($i=0; $i -lt $serviceAccounts.length; $i++){
$rawUserName = $serviceAccounts[$i]
#Benutzer erstellen
Write-Host $rawUserName
New-LocalUser -Name $rawUserName -Password $SecurePassword -FullName $rawUserName -
Description "ServiceAccount"
Set-LocalUser -Name $rawUserName -PasswordNeverExpires 1
#Gruppe setzen
Add-LocalGroupMember -Member $rawUserName -Name Benutzer
Add-LocalGroupMember -Member $rawUserName -Name Administratoren
}
}
```

# Choice1

```
function OutputError([string]$errorMessage)
{
Write-Host $errorMessage -ForegroundColor Red
Read-Host 'Es ist ein Fehler aufgetreten.'
}

$title = 'something'
$question = 'Are you sure you want to proceed?'
$choices = '&Yes', '&No'

#$decision = $Host.UI.PromptForChoice($title, $question, $choices, 1)
$decision = $Host.UI.PromptForChoice('something','Are you sure you want to proceed?',$choices,
1)
if ($decision -eq 0) {
Write-Host 'confirmed'
} else {
Write-Host 'cancelled'
}


$Title = "Title"
$Info = "Pick Something!"
$options = echo Option1 Option2 Option3
$defaultchoice = 2
$selected = $host.UI.PromptForChoice($Title , $Info , $Options, $defaultchoice)
$options[$selected]

$Parent = Read-Host -prompt "Enter full parent path that will contain the new folder"

if ( ($Parent -eq $null) -or ($Parent -eq "") -or ($Parent -eq " ")) { Write-Host "You entered a blank
value: This Script is now Exiting." }
else { Write-Host "You Entered $Parent" }
```

# Choice 2

```
param(
[String] $userName,
[String] $passWord
)


$title = 'something'
$question = 'Are you sure you want to proceed?'
$choices = '&Yes', '&No'

$decision = $Host.UI.PromptForChoice($title, $question, $choices, 1)
if ($decision -eq 0) {
Write-Host 'confirmed'
} else {
Write-Host 'cancelled'
}

function GetStringValue([String]$title)
{
Write-Host $title
$inputString = Read-Host -prompt " -> Ohne Eingabe wird das Script abgebrochen!"
if ( ($inputString -eq $null) -or ($inputString -eq "") -or ($inputString -eq " ")) { return "" }
else { return $inputString }
}

function EnsureStringValueOrAbort([String]$ensureValue, [String]$title )
{
if ( -not $ensureValue ) {
Write-Host $title
$ensureValue = Read-Host -prompt " -> Ohne Eingabe wird das Script abgebrochen!"
if ( ($ensureValue -eq $null) -or ($ensureValue -eq "") -or ($ensureValue -eq " ")) {
Write-Host "cancelled"
exit 1
}
else { return $ensureValue }
}
}

function EnsureStringValueOrDefault([String]$ensureValue, [String]$defaultValue, [String]$title)
{
```

```
if ( -not $ensureValue ) {
Write-Host $title
$ensureValue = Read-Host -prompt " -> Ohne Eingabe wird der Defaultwert verwenden
($defaultValue)!"
if (($ensureValue -eq $null) -or ($ensureValue -eq "") -or ($ensureValue -eq " ")) { return
$defaultValue }
else { return $ensureValue }
if ( -not $ensureValue ) {
Write-Host "cancelled"
exit 1}# Kein Wert also doch abbrechen
}
}


$passWord = EnsureStringValueOrAbort $passWord "Bitte geben Sie ein Passwort an."
Write-Host $passWord

$userName = EnsureStringValueOrDefault $userName "sa" "Bitte geben Sie einen Benutzername
an."
Write-Host $userName
```

# InputSample

```
param(
[Parameter(Mandatory=$true, Position=0)] [String] $userName,
[Parameter(Mandatory=$true, Position=1)] [String] $passWord
)

exit

function GetStringValue([String]$title)
{
Write-Host $title
$inputString = Read-Host -prompt " -> Ohne Eingabe wird das Script abgebrochen!"
if ( ($inputString -eq $null) -or ($inputString -eq "") -or ($inputString -eq " ")) { return "" }
else { return $inputString }
}

function EnsureStringValueOrAbort([String]$ensureValue, [String]$title )
{
if ( -not $ensureValue ) {
Write-Host $title
$ensureValue = Read-Host -prompt " -> Ohne Eingabe wird das Script abgebrochen!"
if ( ($ensureValue -eq $null) -or ($ensureValue -eq "") -or ($ensureValue -eq " ")) {
Write-Host "cancelled"
exit 1
}
else { return $ensureValue }
}
}

function EnsureStringValueOrDefault
{
Param
(
[Parameter(Mandatory=$true, Position=0)]
[string] $EnsureValue,
[Parameter(Mandatory=$true, Position=1)]
[string] $DefaultValue,
[Parameter(Mandatory=$true, Position=2)]
[string] $Title
)
Write-Host "`$EnsureValue value: $EnsureValue -> `$ID DefaultValue: $DefaultValue -> `$Title
```

```
value: $Title"
}

function Get-Something
{
Param
(
[Parameter(Mandatory=$true, Position=0)]
[string] $Name,
[Parameter(Mandatory=$true, Position=1)]
[int] $Id
)
Write-Host "`$Name value: $Name -> `$ID value: $Id"

}

Get-Something -Id 34 -Name "Blah1"
Get-Something "Blah" 345


Function Test([string]$arg1, [string]$arg2)
{
Write-Host "`$arg1 value: $arg1"
Write-Host "`$arg2 value: $arg2"
return "Hello"
}

$x = Test "ABC" "DEF"
Write-Host $x

Write-Host "before EnsureStringValueOrDefault"
$passWord = EnsureStringValueOrDefault "nunun" "hallo" "Bitte geben Sie ein Passwort an."
Write-Host "after EnsureStringValueOrDefault"
Write-Host $userName

exit
Write-Host "before"
$userName = EnsureStringValueOrAbort($userName, "Bitte geben Sie einen Benutzername an.")
Write-Host "after"
Write-Host $userName
```

# Folder

```
#Erstellt Unterordner im aktuellen Verzeichnis
param(
[String] $folderName
)
$exitCode = ":q"

if ( -not $folderName ) {
Write-Host "Geben sie den Verzeichnisnamen an."
while ( -not $folderName ) {
$folderName = Read-Host "Verzeichnis? ($exitCode = Exit)"
}
}
else
{
Write-Host Parameter: $folderName
return 1
}
if ($folderName -eq $exitCode){return -1}

$currentfolder = Get-Location
#Write-Host $currentfolder
$PATH = "$currentfolder\$folderName"
if (!(Test-Path $PATH)) {New-Item -Path $PATH -ItemType Directory}
```

---

```
Join-Path -Path "path" -ChildPath "childpath"
path\childpath

[IO.Path]::Combine('C:\', 'Foo', 'Bar')
```

---

```
Test-Path

if(![System.IO.File]::Exists($path)){
# file with path $path doesn't exist
}
```

# WARNUNG: Es kann kein Download von URI

## Lösung

Eine funktionierende Internetverbindung vorausgesetzt, liegt es vermutlich an dem „Upgrade“ des Azureedges, der den Lookup-Provider stellt (onegetcdn.azureedge.net/providers). Der Endpunkt lässt seit etwa April 2020 keine TLS1.0 Verbindungen mehr zu und wurde auf „TLS1.2 min“ konfiguriert. Die PowerShell (<14393.3474) spricht per Default aber nur TLS1.1.

Dieser PS-Befehl stellt die PowerShell ebenfalls auf 1.2 um:

```
[Net.ServicePointManager]::SecurityProtocol=[Net.SecurityProtocolType]::Tls12
```