#-------------------------------------------------------------------------------------------------#

# Load the SQL Provider and SMO

#-------------------------------------------------------------------------------------------------#

# Check to see if the SQL provider is loaded. If not, load it.

# (Trying to load it after it's already loaded will generate an error)

[String] $IsLoaded = Get-PSProvider |

Select-Object Name |

Where-Object { $\_ -match "Sql\*" }

if ($IsLoaded.Length -eq 0)

{

# ----------------------------------------------------------------------------------------

# This section loads the SQL Server PS Provider. If all you are going

# to use is the provider, you only need the code in this section

# ----------------------------------------------------------------------------------------

# Load some mandatory, global variables that SQL Provider needs

Set-Variable -scope Global -name SqlServerMaximumChildItems -Value 0

Set-Variable -scope Global -name SqlServerConnectionTimeout -Value 30

Set-Variable -scope Global -name SqlServerIncludeSystemObjects -Value $false

Set-Variable -scope Global -name SqlServerMaximumTabCompletion -Value 1000

# Now load the actual snap-ins

Add-PSSnapin SqlServerCmdletSnapin100

Add-PSSnapin SqlServerProviderSnapin100

# If snap ins aren't already loaded, safe bet types aren't either.

# Get the folder where SQL Server PS files should be

# This is the path in the registry to the directory name

$SqlPsRegistryPath = "HKLM:SOFTWARE\Microsoft\PowerShell\1\ShellIds" `

+ "\Microsoft.SqlServer.Management.PowerShell.sqlps"

# Return the value associated with the above path

$RegValue = Get-ItemProperty $SqlPsRegistryPath

# Convert it to an actual directy name using the

# GetDirectoryName method of the System.IO.Path class

$SqlPsPath = [System.IO.Path]::GetDirectoryName($RegValue.Path) + "\"

# Set a path to the folder where the Type and format data should be

$sqlpTypes = $SqlPsPath + "SQLProvider.Types.ps1xml"

$sqlpFormat = $sqlpsPath + "SQLProvider.Format.ps1xml"

# Update the type and format data. Updating if its already loaded won't do any harm.

Update-TypeData -PrependPath $sqlpTypes

Update-FormatData -prependpath $sqlpFormat

# ----------------------------------------------------------------------------------------

# Load the assemblies so we can use the SMO objects

# ----------------------------------------------------------------------------------------

$assemblylist =

"Microsoft.SqlServer.ConnectionInfo",

"Microsoft.SqlServer.SmoExtended",

"Microsoft.SqlServer.Smo",

"Microsoft.SqlServer.Dmf",

"Microsoft.SqlServer.SqlWmiManagement",

"Microsoft.SqlServer.Management.RegisteredServers",

"Microsoft.SqlServer.Management.Sdk.Sfc",

"Microsoft.SqlServer.SqlEnum",

"Microsoft.SqlServer.RegSvrEnum",

"Microsoft.SqlServer.WmiEnum",

"Microsoft.SqlServer.ServiceBrokerEnum",

"Microsoft.SqlServer.ConnectionInfoExtended",

"Microsoft.SqlServer.Management.Collector",

"Microsoft.SqlServer.Management.CollectorEnum"

foreach ($asm in $assemblylist)

{

[void][Reflection.Assembly]::LoadWithPartialName($asm)

}

}

# Normally you wouldn't write anything to the screen, but since

# this is a demo we just want a warm fuzzy its been loaded

Clear-Host

Write-Host "SQL Server Libraries are Loaded"