Practical PowerShell For SQL Server Developers and DBAs

Download the latest version of this PowerShell™ wallchart and read the accompanying in-depth article from Simple-Talk at http://bit.ly/MZmOX9

Module Repositories

Store your modules—including sqlps—in one of these.

system-level	$env:windir \System32\WindowsPowerShell\v1.0\Modules
user-level	$HOME \Documents\WindowsPowerShell\Modules

Modules vs. Snap-ins

Loading snap-ins for SQL support are deprecated, though available.

	SQL Server 2008 & 2008 R2	SQL Server 2012
Cmdlet support	SQLServerCmdletSnapin100	SQLServerCmdletSnapin110
Provider support	SQLServerProviderSnapin100	SQLServerProviderSnapin110

Rather, use the sqlps module instead.

	Load	View items loaded	View items available to load
Modules	Import-Module	Get-Module	Get-Module -ListAvailable
Snap-ins	Add-PSSnapin	Get-PSSnapin	Get-PSSnapin -Registered

Load sqlps module:

```
Import-Module sqlps –DisableNameChecking
```

Executing Queries

Invoke-Sqlcmd

[-ServerInstance <PSObject>]
[-Database <String>]
[-EncryptConnection]
[-Username <String>]
[-Password <String>]
[[-Query] <String>]
[-QueryTimeout <Int32>]
[-ConnectionTimeout <Int32>]
[-ErrorLevel <Int32>]
[-SeverityLevel <Int32>]
[-MaxCharLength <Int32>]
[-MaxBinaryLength <Int32>]
[-AbortOnError]
[-DedicatedAdministratorConnection]
[-DisableVariables]
[-DisableCommands]
[-HostName <String>]
[-NewPassword <String>]
[-Variable <String[]>]
[-InputFile <String>]
[-OutputSqlErrors]
[-SuppressProviderConte xtWarning]
[-IgnoreProviderContext]
[<CommonParameters>]

Profile Locations

Store aliases and commands that you always want available.

all users and **all** shells	$env:windir\System32\WindowsPowerShell\v1.0\profile.ps1
all users and Microsoft.PowerShell shell	$env:windir\System32\WindowsPowerShell\v1.0\ Microsoft.PowerShell_profile.ps1
current user and **all** shells	$HOME\Documents\WindowsPowerShell\profile.ps1
current user and Microsoft.PowerShell shell	$HOME\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1

Determining Default Properties

Nodes reveal only a portion of their properties by default; see **Key Commands** for more.

Type specified in **module-specific** formatting file?	$env:windir\system32\ WindowsPowerShell \v1.0\Modules*. format.ps1xml $HOME\Documents\WindowsPowerShell\Modules*.format.ps1xml
Type specified in **system** formatting file?	$env:windir\system32\ WindowsPowerShell \v1.0*.format.ps1xml
DefaultDisplayPropertySet defined?	Use $*object*.PSStandardMembers.DefaultDisplayPropertySet; otherwise, display all properties

Determining List or Table Output

Nodes display either in PowerShell's list format or table format automatically, unless you explicitly specify otherwise.

Format-verb cmdlet specified in pipeline?	… | Format-List … … | Format-Table …
Type specified in **module-specific** formatting file?	$env:windir\system32\ WindowsPowerShell \v1.0\Modules*. format.ps1xml $HOME\Documents\WindowsPowerShell\Modules*.format.ps1xml
Type specified in **system** formatting file?	$env:windir\system32\ WindowsPowerShell \v1.0*.format.ps1xml
More than four properties?	Use **Format-List**; otherwise, use **Format-Table**

Nodes in SQL Server Space

Description	Node	Default Properties	Object Type (… prefix = Microsoft.SqlServer.Management.)
SQL Server data store root	\	Name, Root, Description	….PowerShell.Extensions.SqlServerProviderExtension
Network root	\SQL	MachineName	….PowerShell.Extensions.Machine
Instances on selected machine	\SQL*machine*	InstanceName	….Smo.Server
Top-level instance objects	\SQL*machine**instance*	***list of object names***	*System.String*
Databases in selected instance	\SQL*machine**instance*\Databases	Name, Status, RecoveryModel, CompatLvl, Collation, Owner	….Smo.Database
Top-level DB objects	\SQL*machine**instance*\Databases*database*	***list of object names***	*System.String*
- - - - Selected database nodes - - - -			
Tables in selected database	\SQL*machine**instance*\Databases*database*\Tables	Schema, Name, Created	….Smo.Table
Views in selected database	\SQL*machine**instance*\Databases*database*\Views	Schema, Name, Created	….Smo.View
Roles in selected database	\SQL*machine**instance*\Databases*database*\Roles	Name	….Smo.DatabaseRole
Triggers in selected database	\SQL*machine**instance*\Databases*database*\Triggers	Name, Created	….Smo.Trigger

Top-level nodes: \SQL, \SQLPolicy, \SQLRegistration, \Utility, \DAC, \DataCollection, \IntegrationServices, \SQLAS

Database nodes: ApplicationRoles, Assemblies, AsymmetricKeys, Certificates, DatabaseAuditSpecifications, Defaults, ExtendedProperties, ExtendedStoredProcedures, FileGroups, FullTextCatalogs, FullTextStopLists, LogFiles, PartitionFunctions, PartitionSchemes, PlanGuides, **Roles**, Rules, Schemas, ServiceBroker, StoredProcedures, SymmetricKeys, Synonyms, **Tables**, **Triggers**, UserDefinedAggregates, UserDefinedDataTypes, UserDefinedFunctions, UserDefinedTableTypes, UserDefinedTypes, Users, **Views**, XmlSchemaCollections

Executing Queries

For brevity, **sql** aliases **Invoke-Sqlcmd**: `New-Alias sql Invoke-Sqlcmd`

Description	Command sequence
Query with default context	`sql -Query "select db_name()"`
Query specifying server	`sql -Query "select db_name()" -Server .\sqlexpress`
Query to interactive grid	`sql -Query "select * from clents" | Out-GridView`
Output in table format	`sql "select * from clents" | Format-Table –AutoSize`
Output in list format	`sql "select * from clents" | Format-List`

Cmdlets implemented by SQL Server Provider

Cmdlet	Canonical alias	Other aliases	Description
Get-Location	gl	pwd	Gets current node
Set-Location	sl	cd, chdir	Changes current node
Get-ChildItem	gci	dir, ls	Lists the objects at current node
Get-Item	gi		Properties of current node
Rename-Item	rni	ren	Renames an object
Remove-Item	ri	del, erase, rd, rm, rmdir	Removes an object

Source: Navigate SQL Server PowerShell Paths on MSDN

Key Commands

For brevity **sandboxDB** aliases **SQLSERVER:\SQL*machine**instance*\databases\sandbox** and **gci** aliases **Get-ChildItem**.

Description	Command sequence
Go to root of SQL Server data store	`Set-Location SQLSERVER:\`
Go to root of DB objects	`Set-Location SQLSERVER:\SQL`
List instance names on local machine	`Get-ChildItem SQLSERVER:\SQL\machine`
List databases on selected instance	`Get-ChildItem SQLSERVER:\SQL\machine\instance\Databases`
List tables in selected database	`gci SQLSERVER:\SQL\machine\instance\Databases\database\Tables`
List tables with shortcut, default properties	`gci sandboxDB:\Tables`
List subset of tables using SMO property	`gci sandboxDB:\Tables | where {$_.Schema -eq "dbo"}`
List tables,all properties	`gci sandboxDB:\Tables | Format-Table –Force *`

SQL Server drives

Use **New-PSDrive** to create drive shortcuts.

```
New-PSDrive –Name DB –PSProvider SQLSERVER
    –Root SQLSERVER:\sql\localhost\SQLEXPRESS\Databases
```

```
Get-PSDrive | ? { $_.Provider.Name -eq "SqlServer" } | select name,root
```

```
Name       Root
----       ----
DB         SQLSERVER:\sql\localhost\SQLEXPRESS\Databases
SQLSERVER  SQLSERVER:\
```

Scripting Tables or Complete Database

For brevity **sandboxDB** aliases **SQLSERVER:\SQL\ *machine**instance*\databases\sandbox**, **%** aliases **ForEach-Object**, and **?** aliases **Where-Object**.

Description	Command sequence
All tables (output to console)	`gci sandboxDB:\Tables | % { $_.Script() }`
All tables (output to file)	`gci sandboxDB:\Tables | % { $_.Script() } | Set-Content C:\create.sql`
All tables (separate batches)	`gci sandboxDB:\Tables | % { $_.Script() + "GO" }`
Selected tables	`gci sandboxDB:\Tables | ? { $_.name -match "big.*" } | % { $_.Script() }`
Single table	`(gci sandboxDB:\Tables | ? { $_.name -eq "xyz_table" }).Script()`

All tables with their indexes	`gci sandboxDB:\Tables | % {`
	`  $_.Script() + "GO"`
	`  $_.Indexes | % { $_.Script() + "GO" }`
	`}`

Full database	`$myScriptFile = filepath`
	`$myDbInstance = Get-Item nodepath`
	`$mydb = $myDbInstance.Databases["dbName"]`
	`$mydb.Script() | Out-File $myScriptFile`
	`$scrp = new-object ('Microsoft.SqlServer.Management.Smo.Scripter') ($myDbInstance)`
	`$scrp.Options.AppendToFile = $True`
	`$scrp.Options.FileName = $myScriptFile`
	# Other options here…
	`$scrp.Script([Microsoft.SqlServer.Management.Smo.SqlSmoObject[]]$mydb.Tables)`

References

Learning PowerShell, PowerShell in SQL Server 2012 / 2008, SQL Server PowerShell Help, Using SQL Server Cmdlets

Key Terms

SMO	.NET classes to create applications that manage SQL Server.
SQLPS	Module providing SQL Server support in PowerShell in the form of new cmdlets (Invoke-Sqlcmd and others) and a SQL Server provider.
SQL Server Provider	Allows you to interact with the hierarchy of SQL Server objects just as a native PowerShell file system provider allows you to interact with files. You can navigate through SQL Server space using paths to nodes analogously to paths to files.
SQLPSX	CodePlex project introduced prior to sqlps to provide SQL Server support in PowerShell, SQLPSX still provides some functionality not covered by sqlps.