**Managing SQL Server Services with PowerShell**

[**http://www.databasejournal.com/features/mssql/managing-sql-server-services-with-powershell.html**](http://www.databasejournal.com/features/mssql/managing-sql-server-services-with-powershell.html)

**By** [**Arshad Ali**](http://www.databasejournal.com/feedback/http:/www.databasejournal.com/features/mssql/managing-sql-server-services-with-powershell.html)

**Introduction**

PowerShell provides a command-line shell and scripting language (built in the .NET Framework) especially designed for administrative task automation and configuration management. PowerShell includes commands (called cmdlets) to administer SQL Server along with having cmdlets for administering other Microsoft server products. This gives administrators a common scripting language across servers’ platforms.

As PowerShell provides many more capabilities than simply administering SQL Server, it’s not possible to cover everything in one article. This article is going to talk about how, with the help of PowerShell, you can manage Windows Services related to SQL Server either on a local machine or remote machine.

**Identifying SQL Server Services**

[Get-Service](http://technet.microsoft.com/en-us/library/hh849804.aspx) is PowerShell cmdlet that retrieves a list of Windows services either from a local machine or from a remote machine.

*Please ensure the PowerShell command prompt or PowerShell ISE tool is launched with Administrator elevated permissions to execute these cmdlets in order to access Windows services and manage it. Also you need to be an administrator on the target machine to manage Windows services.*

# Returns all windows services on local machine

Get-service

# Returns all windows services on "MySQLBox" remote machine

Get-service -computername MySQLBox

# Returns all windows services on local machine which are in running state

Get-service | where-object {$\_.Status -eq "Running"}

**Related Articles**

* [SQL Server 2012 Integration Services - Introduction to Managing SQL Server with PowerShell](http://www.databasejournal.com/features/mssql/ssis-2012-introduction-to-managing-sql-server-with-powershell.html)
* [SSIS 2012 - Using PowerShell to Remotely Execute File System Packages](http://www.databasejournal.com/features/mssql/ssis-2012-using-powershell-to-remotely-execute-file-system-packages.html)
* [SQL Server 2012 Integration Services - Leveraging PowerShell in Package Deployment Model](http://www.databasejournal.com/features/mssql/ssis-2012-leveraging-powershell-in-package-deployment-model.html)
* [SQL Server 2012 Integration Services - PowerShell-Based Project Deployment](http://www.databasejournal.com/features/mssql/ssis-2012-powershell-based-project-deployment.html)
* [SQL Server 2012 Integration Services - Using PowerShell to Configure Package Execution Parameters](http://www.databasejournal.com/features/mssql/ssis-2012-using-powershell-to-configure-package-execution-parameters.html)

You can apply filters with Get-Service cmdlets to retrieve only services that contain some specific words in its name. For example, the script below produces a list of Windows services from the local machine, which contain “SQL” in its name (all the services related to SQL Server).

# Returns all windows services on local machine which contains SQL word in its display name

Get-service \*SQL\*

# Returns all windows services on local machine which contains SQL word in its display name and which are in running state

Get-service \*SQL\* | Where-Object {$\_.status -eq "Running"}

**Starting and Stopping SQL Server Services**

You can use [Start-Service](http://technet.microsoft.com/en-us/library/hh849825.aspx) PowerShell cmdlets to start a Windows service on a local or remote machine. With this script you can start SQL Server related services.

# Start SQL Server Database engine service (default instance)

Start-Service -Name 'MSSQLSERVER'

# Start SQL Server Database engine service (named instance CONTOSO)

Start-Service -Name 'MSSQL$CONTOSO'

# Start SQL Server Integration Services on SQL Server 2012 box

Start-Service -Name 'MsDtsServer110'

# Start SQL Server Integration Services on SQL Server 2008 and 2008 R2 box

Start-Service -Name 'MsDtsServer100'

# Start SQL Server Analysis services engine service (default instance)

Start-Service -Name 'MSSQLServerOLAPService'

# Start SQL Server Analysis services engine service (named instance CONTOSO)

Start-Service -Name 'MSOLAP$CONTOSO'

# Start SQL Server Reporting Server service (default instance)

Start-Service -Name 'ReportServer'

# Start SQL Server Reporting Server service (named instance CONTOSO)

Start-Service -Name 'ReportServer$CONTOSO'

# Start SQL Server SQL Server Agent service (default instance)

Start-Service -Name 'SQLSERVERAGENT'

# Start SQL Server SQL Server Agent service (named instance CONTOSO)

Start-Service -Name 'SQLAgent$CONTOSO'

Likewise, you can use [Stop-Service](http://technet.microsoft.com/en-us/library/hh849790.aspx) PowerShell cmdlets to stop a running Windows service on a local or remote machine.

# Stop SQL Server Database engine service (default instance)

Stop-Service -Name 'MSSQLSERVER'

# Stop SQL Server Database engine service (named instance CONTOSO)

Stop-Service -Name 'MSSQL$CONTOSO'

# Stop SQL Server Integration Services on SQL Server 2012 box

Stop-Service -Name 'MsDtsServer110'

# Stop SQL Server Integration Services on SQL Server 2008 and 2008 R2 box

Stop-Service -Name 'MsDtsServer100'

# Stop SQL Server Analysis services engine service (default instance)

Stop-Service -Name 'MSSQLServerOLAPService'

# Stop SQL Server Analysis services engine service (named instance CONTOSO)

Stop-Service -Name 'MSOLAP$CONTOSO'

# Stop SQL Server Reporting Server service (default instance)

Stop-Service -Name 'ReportServer'

# Stop SQL Server Reporting Server service (named instance CONTOSO)

Stop-Service -Name 'ReportServer$CONTOSO'

# Stop SQL Server SQL Server Agent service (default instance)

Stop-Service -Name 'SQLSERVERAGENT'

# Stop SQL Server SQL Server Agent service (named instance CONTOSO)

Stop-Service -Name 'SQLAgent$CONTOSO'

Stop-Service cmdlets might fail if you try to stop a service on which another service is dependent. For example, SQL Server Agent service is dependent on SQL Server database engine service and hence if you try stopping SQL Server database engine service without first stopping SQL Server Agent service the command will fail. In this case, either you first need to stop SQL Server Agent service then stop SQL Server database engine service or else use *–Force* parameter to forcefully stop all the dependent services.

# Stop SQL Server Database engine service (default instance) along with dependent service

Stop-Service -Name 'MSSQLSERVER' -Force

# Stop SQL Server Database engine service (named instance CONTOSO) along with dependent service

Stop-Service -Name 'MSSQL$CONTOSO' -Force

Please note, based on the SQL Server installation type (default or named) the name of the service may vary and hence you need to provide the right service name when starting or stopping the service name. For example, with default installation service name for SQL Server database engine is MSSQLSERVER but suppose you have a named instance called CONTOSO the service name will be MSSQL$CONTOSO. You can find details about SQL Server services [here](http://blogs.technet.com/b/fort_sql/archive/2010/05/31/list-of-sql-server-service-names.aspx).

**Managing SQL Server Services on a Remote Machine**

As an administrator you might be responsible for maintaining several SQL Server machines. Now consider a scenario where you are required to change the domain user account for SQL Server related services on all of these machines or if you want to update the domain user account or its password for all of these services (on the local or on the remote). Do you really need to log on to each machine and make changes? Not exactly, with the help of PowerShell scripting you not only can start and stop the services but can also change other properties like log-on user account or password, startup type, etc.

With the script below I am first creating a remote session with MySQLBox machine and then executing commands to change the log-on user account name and password.

#Create a new remote PowerShell session and pass in the scrip block to be executed

$session = New-PSSession -ComputerName MySQLBox -Credential Domain01\User01

$UserName = "" # specify user Name here

$Password = "" # specify Password here

Invoke-Command -Session $session -ArgumentList $UserName, $Password -Scriptblock {

       param($UserName, $Password)

    # Start SQL Server Database engine service (default instance)

       $Svc = Get-WmiObject win32\_service -filter "name='MSSQLSERVER'"

       $Svc.Change($Null, $Null, $Null, $Null, $Null, $Null, $UserName, $Password)

       Stop-Service -Name 'MSSQLSERVER' -Force

    Start-Service 'MSSQLSERVER'

    # Start SQL Server Integration Services on SQL Server 2012 box

       $Svc = Get-WmiObject win32\_service -filter "name='MsDtsServer110'"

       $Svc.Change($Null, $Null, $Null, $Null, $Null, $Null, $UserName, $Password)

       Stop-Service -Name 'MsDtsServer110' -Force

    Start-Service 'MsDtsServer110'

    # Start SQL Server Analysis services engine service (default instance)

       $Svc = Get-WmiObject win32\_service -filter "name='MSSQLServerOLAPService'"

       $Svc.Change($Null, $Null, $Null, $Null, $Null, $Null, $UserName, $Password)

       Stop-Service -Name 'MSSQLServerOLAPService' -Force

    Start-Service 'MSSQLServerOLAPService'

    # Start SQL Server Reporting Server service (default instance)

       $Svc = Get-WmiObject win32\_service -filter "name='ReportServer'"

       $Svc.Change($Null, $Null, $Null, $Null, $Null, $Null, $UserName, $Password)

       Stop-Service -Name 'ReportServer' -Force

    Start-Service 'ReportServer'

    # Start SQL Server SQL Server Agent service (default instance)

       $Svc = Get-WmiObject win32\_service -filter "name='SQLSERVERAGENT'"

       $Svc.Change($Null, $Null, $Null, $Null, $Null, $Null, $UserName, $Password)

       Stop-Service -Name 'SQLSERVERAGENT' -Force

    Start-Service 'SQLSERVERAGENT'

}

After changing the log-on user account name and password you need to stop and start or restart the service in order for the change to take effect.

**Conclusion**

PowerShell provides a command-line shell and scripting language (built in the .NET Framework) especially designed for administrative task automation and configuration management. In this article, I demonstrated how, with the help of PowerShell, you can manage Windows services related to SQL Server either on a local machine or on a remote machine.