Powershell and Scheduling (http://balazsberki.com/2016/12/powershell-and-scheduling/)

11th December, 2016 (http://balazsberki.com/2016/12/powershell-and-scheduling/) · Balazs Berki (http://balazsberki.com/author/balazs_berki/) (http://balazsberki.com/2016/12/powershell-and-scheduling/#respond)

With this post I aim to show how to create a Windows Scheduled Task using powershell with the basic modules by using the COM Objects (Schedule.Service) of Windows. In the second part of my post I will show how to use powershell and SQL Server Management Objects (SMO) to create an SQL Agent job with multiple steps using powershell only.

Creating a Windows Scheduled Task with Powershell

By using $env:COMPUTERNAME the Scheduled Task gets created on the host where the script was executed. We can also specify a hostname here like $Hostname = “Host1”. In this case the Scheduled Task is created on the host specified by $Hostname.

```
$Hostname = $env:COMPUTERNAME
```

We are using the built in System user so the task gets executed with LocalAdmin rights

```
$taskRunAsuser = "SYSTEM"
```

The Microsoft Com object Schedule.Service is used to create Scheduled Task object

```
$service = new-object -com("Schedule.Service")
$service.Connect($Hostname)
$rootFolder = $service.GetFolder("\")

$taskDefinition = $service.NewTask(0)
$regInfo = $taskDefinition.RegistrationInfo
$regInfo.Description = 'Description of Scheduled Task'
```

We are adding 2 types of triggers : Start every day (type 2) at 10:00 and On Startup (type 8) with a delay of 15 min. Available trigger types can be found here: https://msdn.microsoft.com/en-us/library/windows/desktop/aa383978(v=vs.85).aspx (https://msdn.microsoft.com/en-us/library/windows/desktop/aa383978(v=vs.85).aspx)

```
$triggers = $taskDefinition.Triggers
$trigger = $triggers.Create(2)
$trigger.StartBoundary = "2010-06-10T10:00:00"
$trigger.DaysInterval = 1
$trigger.Id = "DailyTriggerId"
$trigger.Enabled = $True

$trigger2 = $triggers.Create(8)
$trigger2.Id = "StartUpTriggerId"
$trigger2.Delay = "PT15M"
$trigger2.Enabled = $True
```

Adding an Action Item

```
$command = "c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe"
$CommandArguments = "-file c:\temp\get_process.ps1"
$Action = $taskDefinition.Actions.Create(0)
$Action.Path = $command
$Action.Arguments = $CommandArguments
```

Creating the Scheduled Task

```
try {
  $res = $rootFolder.RegisterTaskDefinition( "Name of Scheduled Task", $taskDefinition, 2,
$taskRunAsuser , $null, 2) | Out-String
} catch {
  Write-Output "Error while creating Windows Scheduled Task: "
  Write-Output $_
}

if ($res -like "*<RegistrationInfo>*") {
  $res
  Write-Output "Windows Scheduled Task created successfully!"
}
```

When executing the code successfully the contents of the $res variable will be an XML object describing the newly created Scheduled Task.

Creating an SQL Agent Job with Powershell and SMO

First we need to load the required assemblies

```
[System.Reflection.Assembly]::LoadWithPartialName( 'Microsoft.SqlServer.ConnectionInfo' ) | Out-Null
[System.Reflection.Assembly]::LoadWithPartialName( 'Microsoft.SqlServer.SMO' ) | Out-Null
```

Then we can establish the connection to the SQL Server by creating the $SMOServer object

```
$instFullName = "Host\TestInstanceName"
$Connection = New-Object Microsoft.SqlServer.Management.Common.ServerConnection ( $instFullName )
$Connection.ConnectTimeout = 5
$SMOServer = New-Object Microsoft.SqlServer.Management.Smo.Server( $Connection )
```

The Job is created by creating a Job object on the local JobServer. A JobServer is a class which refers to the SQL Server Agent and the msdb database. You can find more detailed information on the JobServer Class here: https://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.smo.agent.jobserver.aspx (https://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.smo.agent.jobserver.aspx)

```
$jobname = "TestJob"
$jobdesc = "This is a test Job"
$jserver = $SMOServer.JobServer
$j = new-object ('Microsoft.SqlServer.Management.Smo.Agent.Job') ($jserver, $jobname)
$j.OwnerLoginName = 'sa'
$j.Description = $jobdesc
$j.Create()
$j.ApplyToTargetServer("(local)")
```

$j refers to the Job which we have created by the $j.Create() Statement above. Now we will use this Job Object to create a schedule for the Job. After creating the Schedule with the Agent.JobSchedule Object Type of SMO we are setting the start time of the job with the TimeSpan constructor. Based on the below code the Job Schedule will trigger the Job to be started every day at 18:20:00

```
$SQLJobSchedule = New-Object -TypeName Microsoft.SqlServer.Management.SMO.Agent.JobSchedule -argumentlist $j, "Schedule1"
$SQLJobSchedule.FrequencyTypes = "Daily"
$SQLJobSchedule.FrequencyInterval = 1
$SQLJobSchedule.ActiveStartTimeOfDay = New-Object TimeSpan(18, 20, 00);
$SQLJobSchedule.Create()
```

When creating the steps for the job the job object is used again. For the commands of the tasks I am loading SQL files.

```
$SchemaUpgrade_sql = Get-Content "F:\upgradescripts\upgrade_123.sql"
$js = new-object ('Microsoft.SqlServer.Management.Smo.Agent.JobStep') ($j, 'Step1')
$js.SubSystem = 'TransactSql'
$js.Command = $SchemaUpgrade_sql
$js.OnFailAction = 'QuitWithFailure'
$js.OnSuccessAction = "GoToNextStep"
$js.Create()
```

```
$SchemaCheck_sql = Get-Content "F:\scripts\Check_Schema.sql"
$js = new-object ('Microsoft.SqlServer.Management.Smo.Agent.JobStep') ($j, 'Step2')
$js.SubSystem = 'TransactSql'
$js.Command = "$SchemaCheck_sql"
$js.OnFailAction = 'QuitWithFailure'
$js.Create()
```

After creating the Job, the Schedule and the Steps we can close the connection to the SQL Server.

```
$SMOServer.ConnectionContext.Disconnect()
$Connection.Disconnect()
```


Posted in Powershell (http://balazsberki.com/category/powershell/), Scheduled Task (http://balazsberki.com/category/scheduled-task/) |

« SQL Server Management Studio: scripting does not show Data_Compression by default (http://balazsberki.com/2016/11/sql-server-management-studio-does-not-show-data_compression-by-default/)

Reading the Windows Event Log – Event ID: Problems with Qualifiers (http://balazsberki.com/2017/02/reading-the-windows-event-log-event-id-problems-with-qualifiers/) »

Leave a comment

Your email address will not be published. Required fields are marked *

Comment