**[Adding a T-SQL Job Step to a SQL Agent Job with PowerShell](http://www.sqlservercentral.com/blogs/sql-dba-with-a-beard/2017/02/20/adding-a-t-sql-job-step-to-a-sql-agent-job-with-powershell/)**

Posted on 20 February 2017

FROM: <http://www.sqlservercentral.com/blogs/sql-dba-with-a-beard/2017/02/20/adding-a-t-sql-job-step-to-a-sql-agent-job-with-powershell/?utm_source=SSC&utm_medium=pubemail>

[In my last post](https://sqldbawithabeard.com/2017/02/13/altering-a-job-step-on-hundreds-of-sql-servers-with-powershell/), I explained how to alter an existing job step across many servers. I also had cause to add a T-SQL Job step to a large number of jobs as well. This is how I did it.  
As before I gathered the required jobs using Get-SQLAgentJob command from the sqlserver module which you can get by installing the latest SSMS from <https://sqlps.io/dl>

This code was run on PowerShell version 5 and will not run on PowerShell version 3 or earlier as it uses the where method  
I put all of our jobs that I required on the estate into a variable called $Jobs. (You will need to fill the $Servers variable with the names of your instances, maybe from a database or CMS or a text file

$Jobs = (Get-SQLAgentJob -ServerInstance $Servers).Where{$\_.Name -like '\*PartOfNameOfJob\*' -and $\_.IsEnabled -eq $true}

Then I can iterate through them with a foreach loop

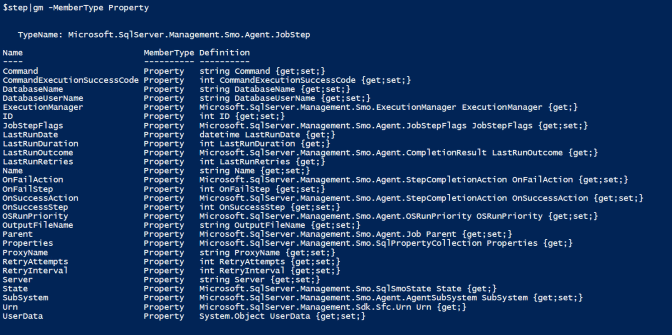
foreach($Job in $Jobs)

Then we need to create a new job step which is done with the following code

$NewStep = New-Object Microsoft.SqlServer.Management.Smo.Agent.JobStep

To find out what is available for this object you can run

$NewStep | Get-Member -MemberType Property



We need to set the name, the parent (The job), the database, the command, the subsystem, the on fail action, on success action and the id for the job step.  
I set the command to a variable to make the code easier to read

$Command = "SELECT Name from sys.databases"

the rest of the properties I fill in inside the loop. To find out what the properties can hold I look at [MSDN for a Microsoft.SqlServer.Management.Smo.Agent.JobStep](https://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.smo.agent.jobstep.aspx)  The ID property is the number of the job step starting at 1 so this example will add a new job step that will be the first to run

$NewStep = New-Object Microsoft.SqlServer.Management.Smo.Agent.JobStep

$NewStep.Name = 'A descriptive name for the job step'

$NewStep.Parent = $Job

$NewStep.DatabaseName = 'master'

$NewStep.Command = $Command

$NewStep.SubSystem = 'TransactSql'

$NewStep.OnFailAction = 'QuitWithFailure'

$NewStep.OnSuccessAction = 'GoToNextStep'

$NewStep.ID = 1

Once the object has all of the properties all we need to do is create it and alter the job

$NewStep.create()

$Job.Alter()

and putting it all together it looks like this

$Jobs = (Get-SQLAgentJob -ServerInstance $Servers).Where{$\_.Name -like '\*PartOfNameOfJob\*' -and $\_.IsEnabled -eq $true}

$Command = "Select name from sys.databases"

foreach($Job in $Jobs)

{

$NewStep = New-Object Microsoft.SqlServer.Management.Smo.Agent.JobStep

$NewStep.Name = 'A descriptive name for the job step1asdfsfasdfa'

$NewStep.Parent = $Job

$NewStep.DatabaseName = 'master'

$NewStep.Command = $Command

$NewStep.SubSystem = 'TransactSql'

$NewStep.OnFailAction = 'QuitWithFailure'

$NewStep.OnSuccessAction = 'GoToNextStep'

$NewStep.ID = 1

$NewStep.create().script()

$Job.Alter()

}

Hopefully this will help you if you need to add a T-SQL Job Step to a large number of servers  
Happy Automating