**Automate SQL Server Administration Tasks Using the Windows Command Prompt**

By: [Jeffrey Yao](https://www.mssqltips.com/sqlserverauthor/121/jeffrey-yao/)   |   Last Updated: 2019-01-22

FROM: <https://www.mssqltips.com/sqlservertip/5888/automate-sql-server-administration-tasks-using-the-windows-command-prompt/?utm_source=dailynewsletter&utm_medium=email&utm_content=headline&utm_campaign=20190122>

**Problem**

Microsoft batch commands have been an existence since the DOS days. With the growth of Windows, most of the commands are enhanced and continue to be an important part of the Windows administration toolset.  If I do not know [PowerShell](https://www.mssqltips.com/sql-server-tip-category/81/powershell/), can I still automate some SQL Server administrations tasks with native Windows commands?  For example, if I want to deploy or execute multiple scripts across multiple server instances, what can I do?

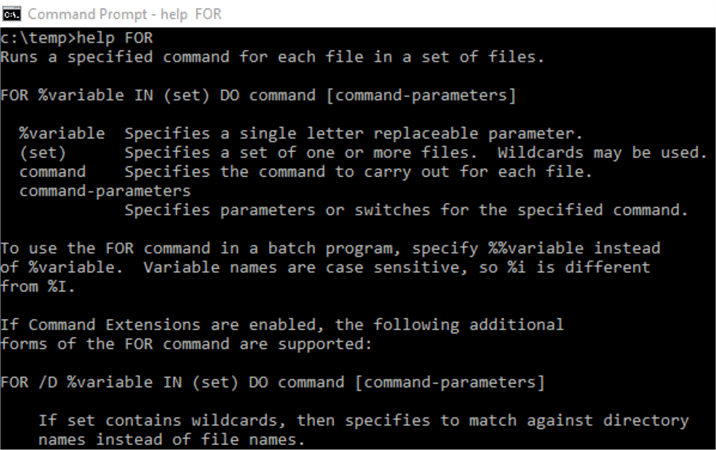
**Solution**

Before we start the solution, we need to give a brief introduction of the Windows command that we need to use for this.  The command is: [FOR](https://ss64.com/nt/for.html).

In Windows 8.1 or Windows 10, we can start a command prompt window by running [cmd.exe](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/cmd) and in the window, type:

help FOR

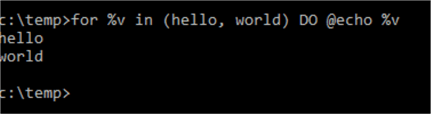
We will get the following:



We can first do a quick test by running the following command:

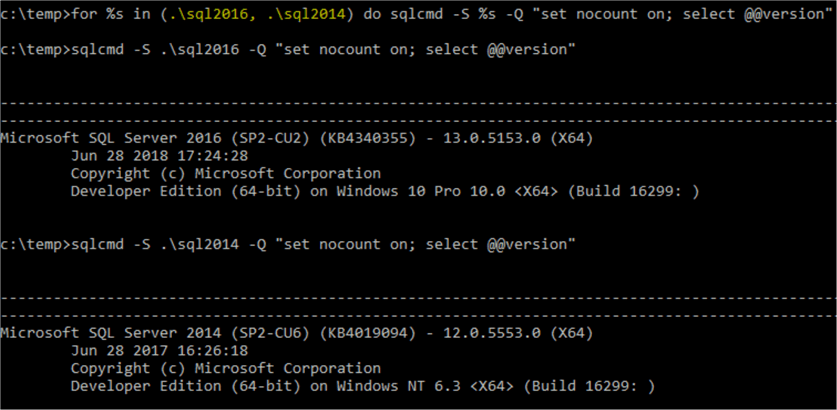
For %v in (hello, world) DO @echo %v

We will get this:



So it is easy to do something like the following: say I want to get the version information of two SQL Server instances.  The code would be:

for %s in (.\sql2016, .\sql2014) do sqlcmd -S %s -Q "set nocount on; select @@version"



Now we will use a few cases to see how we can use the command line commands/tools for real DBA tasks.

Case 1: Execute Full SQL Server Backups of All Databases on an Instance

We can of course write [T-SQL](https://www.mssqltips.com/sqlservertip/1070/simple-script-to-backup-all-sql-server-databases/), with [dynamic SQL statements](https://www.mssqltips.com/sql-server-tip-category/80/dynamic-sql/), to [execute full backups](https://www.mssqltips.com/sqlservertutorial/1/sql-server-backup-options-and-commands-tutorial/) of all databases on a server, but arguably for logic simplicity, we may rely on command line tools to do this task.

The logic is as follows:

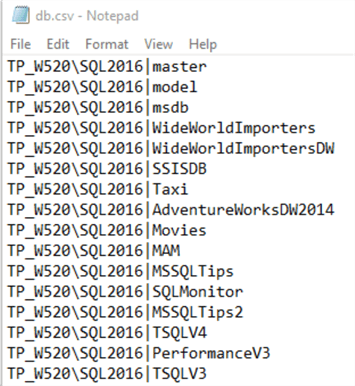
1. Use SQL Server's own tool BCP.exe to dump database names to a text file
2. Use FOR /f to parse the text file from step 1 and then call the sqlcmd.exe to do the backup

The following command will be run from a regular windows command window. It will query against one of my local named instance .\sql2016, and then dump the query result to a csv file as "c:\test\db.csv".

bcp "select ServerName=@@Servername, name from sys.databases where name <> 'tempdb'" queryout "c:\test\db.csv" -S .\sql2016 -T -c -t "|"



If I open this c:\test\db.csv, I will see the following:



[TP\_W520] is my local computer name, so the csv file has basically two columns, separated by a vertical bar |.

The first column has the SQL Server instance name, i.e. TP\_W520\SQL2016, and the second column has the database name.

To read this file c:\test\db.csv, we can use the following command to verify:

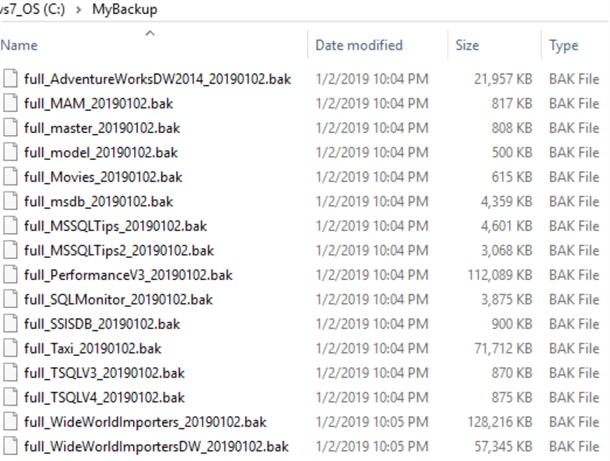
for /f "tokens=1,2 Delims=|" %x in (c:\test\db.csv) do @echo %x %y

Now we can do the backup with the following command (assuming the backup folder is c:\mybackup\):

Set BackupPath=C:\MyBackupSet MyDate=%Date:~10,4%%Date:~4,2%%Date:~7,2%

for /f "tokens=1,2 Delims=|" %%x in (c:\test\db.csv) do sqlcmd -S %%x -Q "backup database %%y to disk='%BackupPath%full\_%%y\_%MyDate%.bak' with init, compression"

We can put all these lines into a batch (or cmd) file, and run the batch file. Once the batch execution finishes, we can check the c:\MyBackup\ folder, and see the backup files generated there:



The second line needs some explanation, it is actually to "cook" the internal environment %Date% into a YYYYMMDD format value. In my environment, %Date% value is like the following:

test

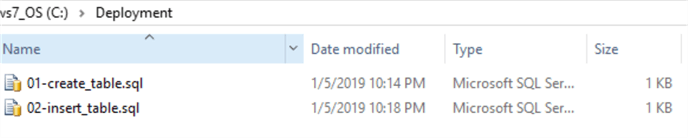
%Date:~m,n% is like a T-SQL substring function, it means from mth position (0 based, i.e. first character is at 0th position), get the following n characters.

So for %Date:~10,4%, it will retrieve 2019. To verify, we can run the following commands:

echo date

Case 2: Run all SQL Server T-SQL Script Files in a Folder Against Multiple SQL Server Instances

There are many cases when we need to run one or multiple scripts under a folder against multiple SQL Server instances. To demonstrate this, we will do the following, under c:\Deployment\ folder, I will have two \*.sql scripts as follows:



The two scripts are:

--01-create\_table.sql

use tempdb

if object\_id('dbo.t', 'U') is not null

 drop table dbo.t;

go

create table dbo.t (id int identity primary key, c varchar(120));

go

--02-insert\_table.sql

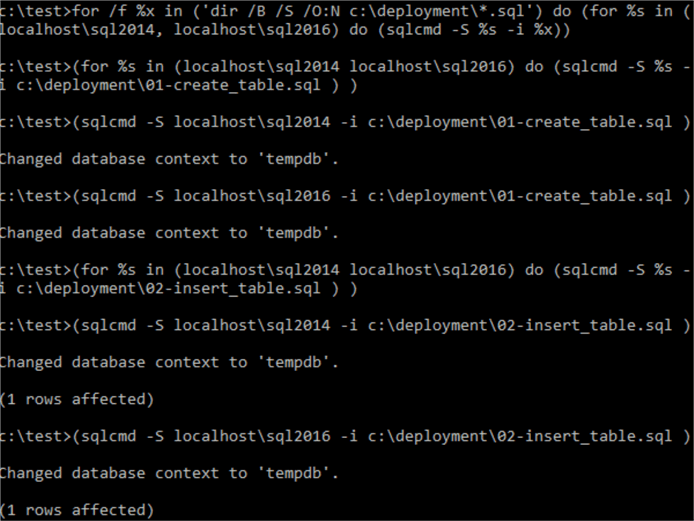
use tempdb

insert into dbo.t (c) values ('hello world');

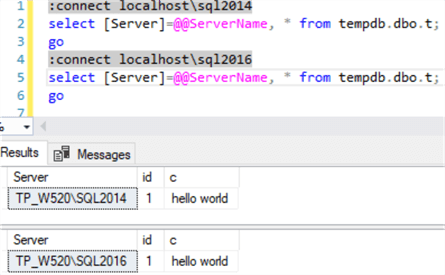
Assume I need to run the two scripts in order, i.e. 01\_xxx.sql should be run before 02\_xxx.sql against two SQL Server instances on my local computer, i.e. localhost\sql2014 and localhost\sql2016, I can use the following command line:

for /f %x in ('dir /B /S /O:N c:\deployment\\*.sql') do (for %s in (localhost\sql2014, localhost\sql2016) do (sqlcmd -S %s -i %x))

This is nested FOR command lines, the first FOR is to list all \*.sql files in C:\Deployment folder, sorted by the name, DIR /O:N means "Order by Name".



In each SQL Server instance, I can see the result as shown below:

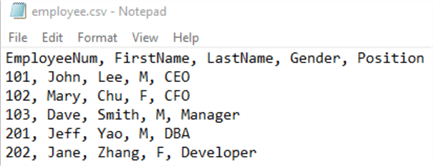


Case 3: Upload portions of a CSV file to a SQL Serve Table

It is pretty easy to use [bcp or bulk insert to do the data import](https://www.mssqltips.com/sqlservertutorial/1/sql-server-backup-options-and-commands-tutorial/), however, if we only want a partial import, i.e. importing some columns of a csv file to a few columns of a target table, we have to use a format file to define the source or target columns we want to use (or skip). You can see details about this topic from [here](https://docs.microsoft.com/en-us/sql/relational-databases/import-export/format-files-for-importing-or-exporting-data-sql-server?view=sql-server-2017) or [here](https://www.mssqltips.com/sqlservertip/1060/bcp-xml-format-files-with-sql-server-2005/).

But in many cases for such partial imports, we may just do it via a simple command line. To demonstrate this, let’s first create a csv file and then a table as follows.

The csv file, c:\test\employee.csv is as follows:



We have a table as follows:

use tempdb

if object\_id('dbo.Employee', 'U') is not null

 drop table dbo.Employee;

go

create table dbo.Employee (

id int identity primary key,

EmployeeNum int,

Position varchar(30),

FirstName varchar(50),

LastName varchar(50),

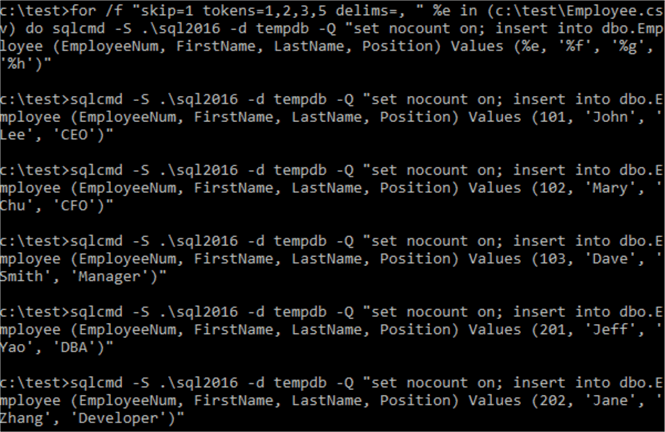
LogDate DateTime default getdate());

So assume we only need to import EmployeeNum, FirstName, LastName and Position columns into the dbo.Employee table, we can do so as follows:

for /f "skip=1 tokens=1,2,3,5 delims=, " %e in (c:\test\Employee.csv) do sqlcmd -S .\sql2016 -d tempdb -Q "set nocount on; insert into dbo.Employee (EmployeeNum, FirstName, LastName, Position) Values (%e, '%f', '%g', '%h')"

The command skips the first line, i.e. the header line of the csv file by using skip=1, and it then picks column 1, 2, 3 and 5 (i.e. EmployeeNum, FirstName, LastName, Position) from the csv file (skipping the 4th column, i.e. Gender column) by using tokens=1,2,3,5 and then puts them into the Value clause of the [Insert SQL statement](https://www.mssqltips.com/sqlservertutorial/2514/sql-server-insert-command-tutorial/) using %e, %f, %g and %h.

After running this command, I can see the screen result is like this:



And checking the table, we can see the data:



Of course, this method may not be the best way if you need to insert thousands of records as this method is basically doing multiple inserts instead of using the more efficient bulk insert. But it is really convenient when the workload is not big and you can type out the command quickly and see each insert executed.

Summary

In this tip, we have demonstrated how we can do some common DBA tasks by just using native Windows commands and SQL Server native tools (bcp.exe and [sqlcmd.exe](https://www.mssqltips.com/sqlservertip/4923/introduction-to-sql-servers-sqlcmd-utility/)). Actually, Windows OS already has an abundance of useful command line tools, such as service control/management utility, [sc.exe](https://support.microsoft.com/en-ca/help/251192/how-to-create-a-windows-service-by-using-sc-exe), multiple file process utility [forfiles.exe](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/forfiles), and many others. Another very useful and new cross-platform utility provided by Microsoft is [mssql-scripter](https://github.com/Microsoft/mssql-scripter), actually Microsoft has released a command line cross-platform query tool called [mssql-cli](https://docs.microsoft.com/en-us/sql/tools/mssql-cli?view=sql-server-2017) as well.

I am a big fan of PowerShell, but I still find many existing Windows command tools can be valuable in handling scenarios that may otherwise need specific knowledge or skills, for example, if I want to see who is currently logged into a remote server that I need to reboot, I will just use the [Query User](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/query-user) command, simple and robust.

When we combine all these existing tools via batch files, we can achieve lots of things without needing to reinvent another tool via either PowerShell or CLR functions.

**Next Steps**

As a DBA, we do not work with SQL Server alone, we actually work with the whole environment where our SQL Server instances reside. Windows OS commands are in fact the basic essential tools that we can use to manage the environment. So to be able to use these tools, and especially those that help us most for DBA tasks, this should be a must have skill for DBAs.

You can review the following links to get more familiar with various Windows commands.

* [Windows Commands References](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands)
* [DOS Commands for the SQL Server DBA](https://www.mssqltips.com/sqlservertip/1729/dos-commands-for-the-sql-server-dba/)
* [Windows FOR command](https://ss64.com/nt/for_f.html)
* Try to simplify the case 1 solution without using BCP to generate a csv file, but just using a nested For command to achieve the same result.

Last Updated: 2019-01-22

|  |  |
| --- | --- |
| **Wednesday, January 23, 2019 - 2:15:38 PM - jeff\_yao** | [Back To Top](https://www.mssqltips.com/sqlservertip/5888/automate-sql-server-administration-tasks-using-the-windows-command-prompt/?utm_source=dailynewsletter&utm_medium=email&utm_content=headline&utm_campaign=20190122#top) |
| Thanks @Kenneth for your kind words. To make good use of the "old" stuff is always fun, and actually native batch scripting (with the help of various existing command line utilities) is more capable than we may be aware of, while there is almost no learning curves (even compared to learning Powershell) | |

|  |  |
| --- | --- |
| **Wednesday, January 23, 2019 - 10:13:16 AM - Kenneth Igiri** | [Back To Top](https://www.mssqltips.com/sqlservertip/5888/automate-sql-server-administration-tasks-using-the-windows-command-prompt/?utm_source=dailynewsletter&utm_medium=email&utm_content=headline&utm_campaign=20190122#top) |
| Really eye-opening, Jefferey. Thanks | |