**Adding more functionality to SQL Server BCP with PowerShell**

By: [Jeffrey Yao](https://www.mssqltips.com/sqlserverauthor/121/jeffrey-yao/)

FROM: <https://www.mssqltips.com/sqlservertip/4354/adding-more-functionality-to-sql-server-bcp-with-powershell/>

**Problem**

As a SQL Server DBA, I frequently need to use the SQL Server [BCP](https://msdn.microsoft.com/en-CA/library/ms162802.aspx) utility to export query data to a CSV file and then pass to other users or applications. However, I find there are some deficiencies with the BCP utility as listed below:

1. It is hard to put double (or single) quotes on each column value.
2. It is hard to add column names as the first line in the exported file.
3. If a column of char[N] (assuming N > 1) has any empty string, i.e. [col]='', BCP will generate [N] blank spaces (when the business requirement is no space).
4. If a column of varchar[N] (assuming N > 1) has any empty string, i.e. [col]='', BCP will generate [1] blank space instead of a real empty string.

Problem Demonstration

The following code will generate a table with 3 rows on my local SQL Server instance, and then I will use bcp.exe to export the SQL Server table to a text file.

Use MSSQLTips -- my test database

--drop table dbo.t;

create table dbo.t (id int identity, b varchar(10), c char(10), note varchar(100));

go

-- populate with 3 records

insert into dbo.t (b, c, note)

select '', '', 'both b and c columns are empty strings, i.e. b='''' and c='''''

union all

select null, null, 'both b and c columns are null, i.e. b=null and c=null'

union all

select 'Hello', ' World', 'both b and c columns have values, for c column, there is leading blank space';

go

Now in an Command window, we run this bcp.exe command

bcp.exe "select \* from dbo.t" queryout "c:\temp\test.txt" -t "|" -S localhost -d MSSQLTips -T -c ;

If I open "c:\temp\test.txt" with NotePad, I will see the following (I highlighted the blank spaces of interest)

1| | |both b and c columns are empty strings, i.e. b='' and c=''

2|||both b and c columns are null, i.e. b=null and c=null

3|Hello| World |both b and c columns are normal with values

The first line demonstrates the issues. For column [b], it is exported to be 1 blank space though it is an empty string, while for column [c] it has 10 blank spaces, though it is an empty string.

Actually, if you look at line 3, you will see World , with leading/trailing blank spaces (the total string is 10 characters), this is because column [c] is defined as char(10), but in many cases users want the trailing spaces trimmed.

**Solution**

To solve the above mentioned issues, I wrote a PowerShell (PS) function to replace BCP utility's export function (with -c parameter). But make no mistake, this is **not** a replacement of the BCP utility, not even with any intention or attempt to do so. But it indeed saves me lots of time and effort to meet user requirements.

<#

.Synopsis

Bulk Copy Query Data to CSV file

.DESCRIPTION

Bulk Copy Data from a query against a SQL Server Database into a CSV file

.EXAMPLE

bcp-data -ServerInstance localhost -Database mssqltips -Query "select \* from dbo.MyTable -FilePath c:\temp\mytable.txt -Trim Y

.INPUTS

None

You cannot pipe objects into BCP-Data

.OUTPUTS

No output

.NOTES

ver 1.0 - Jeffrey Yao 2016/Jun/01

#>

#Requires -Version 3.0

add-type -AssemblyName "System.Data";

function BCP-Data

{

[CmdletBinding()]

Param

(

# SQL Server Instance name with default to local servername

[Parameter(Mandatory=$false)] [string]$ServerInstance=$env:ComputerName,

# Database name with default to 'master'

[Parameter(Mandatory=$false)] [string] $Database='master',

# Query, the query will return a result set

[Parameter(Mandatory=$true)] [String] $Query,

# FilePath, the full path name

[Parameter(Mandatory=$true)] [String] $FilePath,

# FieldTerminator, the separator between columns, default to "|"

[Parameter(Mandatory=$false)] [String] $FieldTerminator="|",

# RowTerminator, the separator between rows, default to a new line "`r`n"

[Parameter(Mandatory=$false)] [String] $RowTerminator="`r`n",

# QuoteMark, the quoting marks around columns, default to none

[Parameter(Mandatory=$false)] [String] $QuoteMark="",

# NoHeader, a switch parameter to decide whether column header are included in the csv file, by default, there is always a header unless this switch parameter is present

[Parameter(Mandatory=$false)] [switch] $NoHeader,

# Trim, decides whether to trim the column of string data type, N = no trim, Y=trim both left and right of the string, L=left trim, R=right trim

[Parameter(Mandatory=$false)]

[ValidateSet('N', 'Y', 'L', 'R')] [String] $Trim='N'

)

#we assume use windows authentication, otherwise, you need to provide User and PassWord for connection

$conn = new-object System.Data.SQLClient.SqlConnection ("server=$ServerInstance; database=$Database; trusted\_connection=true");

$sqlcmd = New-Object System.Data.SqlClient.SqlCommand($query, $conn)

$conn.Open();

$dr = $sqlcmd.ExecuteReader();

$dt = New-Object System.Data.DataTable;

$dt.Load($dr);

$sw = new-object System.IO.StreamWriter($FilePath, $false);

#we first write the header

if (-not $NoHeader)

{

[string]$head=$dt.Columns.columnname -join $FieldTerminator;

$sw.Write($head);

$sw.write($RowTerminator);

}

foreach ( $r in $dt.rows)

{

[string]$tf = $FieldTerminator;

for ([int]$i=0; $i -lt $dt.Columns.count; $i++)

{

if ($i -eq $dt.Columns.count -1 ) #last column does not need to be followed by $FieldTerminator

{ [string]$tf='';}

if ($r[$i] -ne [System.DBNull]::Value)

{

if ($r[$i].GetType().name -ne 'Boolean')

{

$col\_val = $r[$i].ToString() ;

}

else

{

if($r[$i] -eq $true) { $col\_val= '1' } else {$col\_val='0'}

}

$col\_val = $QuoteMark + $(

switch ($trim)

{

'Y' { $col\_val.Trim() }

'N' { $col\_val }

'R' { $col\_val.TrimEnd()}

'L' { $col\_val.TrimStart()}

}

) + $QuoteMark + $tf;

$sw.Write($col\_val);

}

else

{ $sw.Write($QuoteMark+$QuoteMark+$tf);}

} #loop through columns

$sw.Write($RowTerminator);

}#loop through rows

#clean up

$sw.Close();

$dr.Close();

} # bcp-data

Examples

Now, I still use the previously created table and demo the use of different combinations of the function parameters

For demo purposes, I started the PS ISE and opened two windows. In the first window, I copy/pasted the BCP-Data script and ran it and in the second window copy/pasted the following script and ran it to get all the txt files and open them with notepad.exe.

#0. using bcp utility

bcp.exe "select \* from dbo.t" queryout "c:\temp\Test\_bcp.txt" -t "|" -S localhost -d MSSQLTips -T -c

#1. same as previous bcp utility, but corrects single space for empty string issue (to b varchar(10))

bcp-data -ServerInstance localhost -Database MSSQLTips -FilePath "c:\temp\test\_ps.txt" -Query "select \* from dbo.t " -FieldTerminator "|" -NoHeader;

#2. adding header

bcp-data -ServerInstance localhost -Database MSSQLTips -FilePath "c:\temp\test\_header.txt" -Query "select \* from dbo.t " -FieldTerminator "|" ;

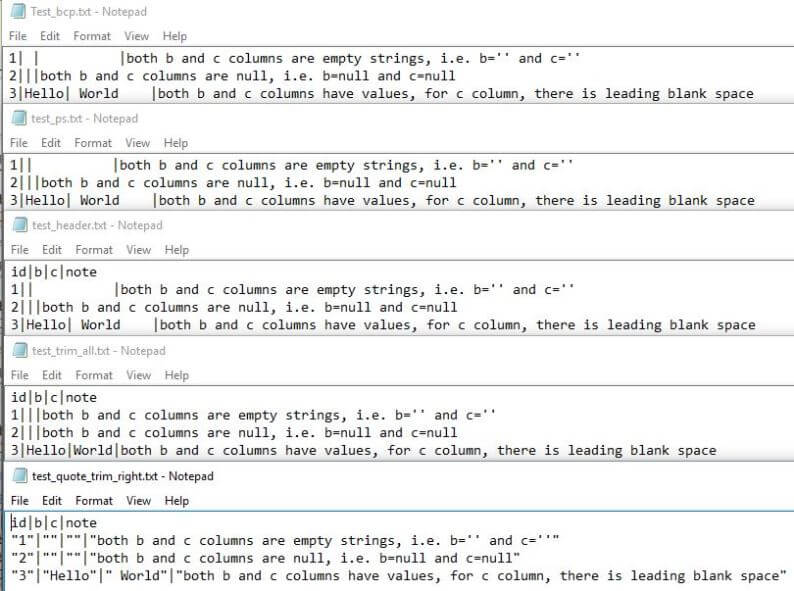
#3. trimming ALL blank spaces

bcp-data -ServerInstance localhost -Database MSSQLTips -FilePath "c:\temp\test\_trim\_all.txt" -Query "select \* from dbo.t " -FieldTerminator "|" -Trim Y; #Y=trim ALL spaces

#4. Adding double quotes to field and also trimming all right spaces

bcp-data -ServerInstance localhost -Database MSSQLTips -FilePath "c:\temp\test\_quote\_trim\_righ.txt" -Query "select \* from dbo.t " -FieldTerminator "|" -Quote '"' -Trim R; #R=Trim Right spaces

Here is the result in sequence:



Summary

In this tip, a new PowerShell function is created to address some weakness in the original BCP.exe utility. Since this is not a compiled utility tool, the performance of this function is 30+ times slower than the BCP utility when big chunks of data need to be exported. For example, for about 20,000 rows in table [AdventureWorks2012].[Person].[Person], using the BCP utility in my environment it takes 0.6 seconds, while using bcp-data function it takes 18 seconds.

So if you are not concerned about performance, or your business requirements cannot be met due to BCP utility's "issues", the bcp-data function is a convenient alternative.

In some rare cases where you do not have bcp.exe installed, you can still use this PS function as it only relies on .Net framework 2.0+, more precisely, System.Data.dll.

**Next Steps**

* You can include this function in your own PS module and play with it.
* You can enhance this function by trying to implement all functions that BCP utility has (even though the performance may not match BCP)
* You may look at the following articles to learn more about data exporting and potential challenges.
  + [Simple way to export SQL Server data to Text Files](https://www.mssqltips.com/sqlservertip/1633/simple-way-to-export-sql-server-data-to-text-files/)
  + [Export data from SQL Server to Excel](https://www.mssqltips.com/sqlservertip/1202/export-data-from-sql-server-to-excel/)
  + [BCP XML Format Files with SQL Server 2005](https://www.mssqltips.com/sqlservertip/1060/bcp-xml-format-files-with-sql-server-2005/)
  + [Possible Solution from StackOverFlow](http://stackoverflow.com/questions/23226847/getting-column-names-with-bcp-queryout)
  + [Other Similar Questions from StackOverflow](http://stackoverflow.com/search?q=bcp+quote)

Last Update: 6/29/2016   
  
  
[**View all my tips**](https://www.mssqltips.com/sqlserverauthor/121/jeffrey-yao/)

* [More Database Developer Tips...](https://www.mssqltips.com/sql-server-developer-resources/)