# SE Consolidation Data Collection

Edited by: Larry Chesnut Version: 4.1.7d Edited: 3/11/2014

# Introduction

The purpose of this document is to describe the steps required to be executed to collect the Consolidation Data.

# Prerequisites

A working knowledge of the following tools is required:

* SQL Server
* Performance Monitor
* PowerShell 3.0

Microsoft Excel

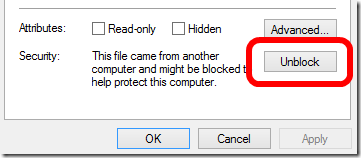
# Data Collection Step-by-Step

Facilitator Note:

1. Setting up the repository server

Grab the latest zipped version of the code from the SE SharePoint before going on site.

When you arrive on site then copy the zipped code to the collecting server, but don't extract the ZIP file yet! Sometimes (not always) the file might need to be unblocked or PowerShell gets cranky because it's considered untrusted. To unblock a file, navigate to it in Windows Explorer, right click, and choose the Properties menu option. On the General tab, click the Unblock button, then click the OK button to close the Properties dialog. Once the file is unblocked you can extract the contents.



Start by creating a **CONSOLIDATION** directory of the customer’s repository server and then a **PowerShellScripts** subdirectory, then copy the SE consolidation package to the **PowerShellScripts** subdirectory. The **CONSOLIDATION** directory is where all the data will eventually be pulled onto. Then un-compress the package. Now let’s see what’s inside.

* **prepConsolidation.ps1** the role of this script is to setup **PerfMon** data collection and repository folders.
* **getSQLinfo.ps1** the role of this script is to gather the **metadata** of both the Windows Server as well as SQL Server and its databases. **NOTE**: adding **Instancename\_PID.csv** output needs PS 3.0 on the **collector** **server**.

**prepConsolidation.ps1** script’s purpose is twofold; build the directory structure in which, PerfMon as well as metadata is collected and stored, and second is the creation of a series of batch scripts that will later be ran to create the PerfMon data collectors on all the target servers. The **prepConsolidation.ps1** script requires an input file named **servers.txt**. The format for line entries is **always NetBiosMachineName\InstanceName**

Here is an example for two standalone instances: SEPCHP7NW\SQL2008R2

SEPCHP7NW\MSSQLServer

**getSQLinfo.ps1** script’s purpose is to collect metadata about the MACHINE\OS\SQL\Databases. It also uses an input file named **server.txt.** Its format for line entries is also **NetBiosMachineName\InstanceName if** the SQL Server instance is not on a ***Windows Clusters***. If it is, then for those clustered SQL instances the line format is a little different as its **VirtualServerName**\InstanceName

In a homogenous environment (all instances are standalone) we will need just one folder named **PowerShellScripts**, and it will have **prepConsolidation.ps1,** **getSQLinfo.ps1,** and a **server.txt** file. But if there is a mixture of standalone and clustered SQL instances, then it just got a little more complicated.

For a **mixed environment** of **standalone** and **clustered instances**, then you will require a separate input file for **getSQLInfo.ps1** script from that used by the **prepConsolidation.ps1** script. We require different files because problems were encountered at various customers when surveying SQL Server instances that were clustered while others were standalone. There have been two approaches to this problem tried in the past; 1) use 2 different input files, (**servers.txt** and **serversGetSQLInfo.txt)**, or 2) use two different directories and separate **prepConsolidation.ps1** script from the **getSQLInfo.ps1** script, and have different input **servers.txt** files in each folder’s PS script.

1. If you have a mixture of clustered and un-clustered SQL Servers, then you can manage the situation by having two different input files, one strictly for the **prepConsolidation.ps1** script in the required format of **NetBiosMachineName\InstanceName** and name it **servers.txt** files. The second file is named **serversGetSQLInfo.txt** and will have a mixture of ***netbiosname\instancename*** and ***virtualservername\instancename*.** You will first run the **prepConsolidation.ps1** script with just ***netbiosname\instancename*** in the **servers.txt** file. Then before running the **getSQLInfo.ps1** script you will have to first edit a line in the script ($serversFile = $scriptsFolder + "servers.txt") to point to **serversGetSQLInfo.txt** file name and then execute the **getSQLInfo.ps1** script.
2. Some have also used the approach of dealing with heterogeneous environment by first creating homogenous sub-group of servers. For each of these homogenous sub-groups a separate set of folders need to be used. For Example: In case of clustered servers, ***netbiosname\instancename*** will be used to collect PerfMon data, but to collect SQL Server and database metadata the ***virtualservername\instancename*** needs to be used. So here you can create 2 separated folders, one dedicated just for clustered servers where the powershellscript sub-folder will just hold the **getSQLinfo.ps1** script and the ***virtualservername\instancename*** formatted **servers.txt** input file reside. The other sub-folder with its own \powershellscript a sub-folder would be for with **prepConsolidation.ps1**, **getSQLinfo.ps1** and **server.txt**file.

Now we can set two separate **server.txt** files for another reason. Another grouping example might be **C:** vs. the **D:** drives, for collecting the PerfMon data on the local servers.

**WARNING**: there must NOT be any white spaces at the end of a line, tabs, carriage returns, or any other gaps in the data of this input file.

In the **PowershellScripts** *folder*, paste the list of servers & respective instances into the **servers.txt** file. The various ways in which this can be done is shown in **sample\_of\_servers.txt** file.

* 1. **WARNING**: Exclude all the servers that cannot use the values that will be in the **prepConsolidation.ps1** script like, the entry **perfMonLocalOutputFolderName** = "C:\CONSOLIDATION\"

This is because some servers may not have **enough disk space on their C**:\ drives.

In order to deal with the exception servers, you will first have to complete the entire set of processes in the document. Then, rename the **servers.txt** file to something like **servers-Cdrives.txt**, and then create a new **servers.txt** file with the list of exception servers that can share a common drive and directory path for storing the PerfMon log and other output. Then, you will run through this document again.

The same consideration for severs in different time zones, maybe domains, etc.

* 1. The directory C:\CONSOLIDATION\ has to have already been created on the target servers.

1. Edit **prepConsolidation.ps1**

Prepare the Consolidation Data Collection Environment

* 1. In the folder, right-click edit prepConsolidation.ps1
  2. Change the **$mainFolder** path in the file to the directory, where “PowershellScripts” folder is stored *on the repository server*. Example …

**$mainFolder = "C:\CONSOLIDATION\"**

* 1. Change the following values accordingly:

$isCollectionOnCentralServer = 0

*0 for collecting PerfMon data locally*

*1 for collecting PerfMon data on repository server*

**NOTE**: Centrally collecting files is prone to various problems. What I have seen happen on several occasions is when more than 10 collections are ran then the central server becomes a bottleneck. If you want to try centrally collecting, you are welcomed to, but don't forget how much time you have to get the collection done. If it stops after 5 or 10 hours, you would have to start all over again☹.

**$perfMonLocalOutputFolderName = "C:\CONSOLIDATION\"**

*Where do you want to store the PerfMon file on the local server?*

**$perfMonNetworkOutputFolderName = "C$\CONSOLIDATION\"**

*Where do you want to store the PerfMon file in the network?*

**$perfmonStartTime = "5/23/2011 02:00:00PM"**

*PerfMon data collection start time.* ***WARNING****: Make sure the start time is scheduled well after the execution of this file. Recommend a least 30 minutes ahead of you executing this, just don’t take too long editing this file.*

**$perfmonEndTime = "5/30/2011 02:00:00PM"**

*PerfMon data collection end time*

**$perfmonSamplingInterval = "00:05:00"**

*Sample interval – HH:MM:SS* ***WARNING****:* Long collections (a week or more) causes the PerfMon log size to get **HUGE!** For a frame of reference, a 5 minute sample interval will generate about 55 MB/day, and if logging lasts for 14 days, then somewhere around 800MB is what you can expect to see. In contrast, a 1 minute interval makes roughly 240 MB/day. The size varies due to the # of: SQL instances and NICs on a given machine. This example had 4 instances of SQL Server and 8 virtual and physical NIC’s.

**$perfmonMaxSize = "100"**

*PerfMon file max size.* ***WARNING****: This parameter may not always work, especially on Win 2003, but* ***lots of problems in general****. Watch out for it looking like it started collecting, but stopping minutes later.*

* 1. Copy the path supplied to $mainFolder line of code. After you close this file, you will need to paste the path in the next file you are about to edit.
  2. Save & close the file

1. Edit **getSQLInfo.ps1**
   1. From the folder, right-click edit **getSQLinfo.ps1**
   2. Paste the path that we copied from **$mainFolder** field of **prepConsolidation.ps1** file into the **$mainFolder** path of the **getSQLInfo.ps1** file
   3. Save & close the file
2. **Execute PowerShell scripts**
   1. Navigate to Start 🡪 All Programs 🡪 Accessories 🡪 Right-Click the Command Prompt Icon and Select “**Run as administrator**”. This step is critical, without doing this you will fail.
   2. In the command prompt type **PowerShell** and hit enter to run PowerShell. You should now see the PS prompt
   3. Execute the following command

**Set-ExecutionPolicy RemoteSigned**

* 1. Change the directory to “C:\Consolidation\PowershellScripts” where the package was copied to.

**CD Consolidation\PowershellScripts**

* 1. Execute the following command

**.\prepConsolidation.ps1**

*This will create on the repository server sub-folders in the main folder named – “Data”, present in the path you edited for the line* ***$mainFolder****. Also it will create additional scripts in “PerfmonTemplates” folder (on the repository server) to start PerfMon data collection.*

* 1. Next, execute this command script in the already open Command Prompt window (command prompt should be in PowerShell mode)

**.\getSQLInfo.ps1**

**WARNING:** *If the first running was with a* ***servers.txt*** *file that held Windows clustered server names, then before executing* ***getSQLinfo.ps1****, be sure you follow the guidance earlier in this document, page 2.*

On Feb 17, 2014 we added additional code to this script so that it would produce at the root of each server name directory a file that has SQL Server instance name, and the PID for that instance. Don’t worry if the rows repeat themselves. The SSIS package (BatchLoadPowerShellData\_V6.dtsx) will later remove duplicate rows and strip out the MSSQL$ from instance names before inserting into the table [dbo].[PID-InstanceName].

* 1. Exit from PowerShell in command prompt by typing “exit”

1. **Execute Command Prompt Scripts**

*Here is a warning for the sake of clarity as you read about the next few steps. Below are references to file names that are not part of the zip file you downloaded from SE’s SharePoint. Do not worry, the file names you will see next will have been created by the above actions.*

*You will also need to have LOCAL ADMIN rights on all the servers you are targeting.*

* 1. Open command prompt (Start🡪All Programs 🡪 Accessories 🡪 Command Prompt (cmd) 🡪 right click and select “Run as administrator”).
  2. Navigate to “**C:\Consolidation\Data\PerfMonTemplates**” directory & run **create\_perf.bat**. The results will be directed to the **CREATE\_OUTPUT.txt** file.
     + You may not need to run these set of scripts if the TASK MANAGER service is running on **ALL** the target servers. **Unfortunately, you cannot always count on the service running** **on ALL the servers**, and so these scripts will initiate everything for you. This script is also useful if you want to trigger the PerfMon data collection forcibly. For ex. You want to start the PerfMon counters right away instead of the time supplied to the **$perfmonStartTime** field in the above step - 2. (C); in such cases, you run **start\_perf.bat**. The results will be directed to the **START\_OUTPUT.txt**
     + You may find in the **CREATE\_OUTPUT.txt** file this message; Error: The information specified does not resolve to a valid path name. A customer hit this repeatedly, but only on his Win2008 machines, 2003 were fine. Problem is permissions. If working from an XP workstion, this can be resolved by copying the PerfMon template file PS had created for the affected server\instance and running the “**logman create counter …”** command locally.
  3. 15 - 25 minutes after starting the PerfMon collection, you will need to run a script found in the “PerfMonTemplates” directory of the repository server, named **dir\_perf\_folder.bat**. The output will be directed to the **DIR\_OUTPUT.txt** file. This will give you the details of the PerfMon files with their sizes. If the file sizes are growing this is a way for you to tell if the data is being collected. Bottom-line, verify if the **\*.blg** files are generated and are more than 320KB. This script is specifically useful when collecting the data locally (on each of the remote servers).
  4. **stop\_perf.bat** is another script available to **if** you want to **stop** the **PerfMon** data **collection** **forcibly**. ***Most of the time, you will never use this script***, but we supply it for the “just in case situation”, say for instance, you want to stop the PerfMon counters right away instead of the time supplied to the **$perfmonEndTime** field in the above step - 2. (C); in such cases, you run **stop\_perf.bat**. The results will be directed to the **STOP\_OUTPUT.txt**
  5. Move the collected files to a central location: run **move\_perf.bat**. The results will be directed to the **MOVE\_OUTPUT.txt**. The PerfMon files will be moved to the central server in the “Data” folder of the $mainFolder path. Verify this by going to the PerfMon folder of a few random servers
  6. Run **delete\_perf.bat** & **delete\_perf\_folder.bat** to clean up the client server. The results will be directed to the **DELETE\_OUTPUT.txt** & **DELETE\_FOLDER\_OUTPUT.txt**. This will delete the data collector set & the data collection folders made by our scripts on the local or central server.

1. **Optional - Run SQL Data Script**

Navigate to $mainFolder & further to SQLScript folder, run **run\_sql.bat**. This will collect additional SQL information. This data is usually captured only when Architect specifically mentions to get it.

1. **FTP the data**

This last step you may not perform for quite a while, either 24 hours, 1 week, or even month later, depending on the customer and engagement situation.

* 1. Once data collection has started, contact the Back Office team using our SharePoint HelpDesk page to submit a service request to setup an FTP site. Someone will send you back a link, login, and password information.
  2. Zip the single Data Folder
  3. Ship it to the FTP created by SE for the engagement
  4. Email the particular SE or SES team members responsible for this engagement (the people may change) and let them know the data is ready for downloading and processing.