|  |  |
| --- | --- |
| **Automate Your Central Management Server Registrations**  **By** [**S. Kusen**](http://www.sqlservercentral.com/Authors/Articles/S._Kusen/383329/)**, 2014/08/18**  **FROM:** [**http://www.sqlservercentral.com/articles/Administration/113350/**](http://www.sqlservercentral.com/articles/Administration/113350/)  **Introduction**  The Central Management Server (CMS) is a feature that was introduced with SQL Server 2008 and has not really changed much through SQL 2012. I have not reviewed it for 2014 yet, and the scope of this article will only go up to 2012.  The CMS allows for a single instance to be able to manage multiple servers, and you can group those servers as you see fit.  What I will show in this article is how to leverage the stored procedures within the MSDB database and an existing inventory of SQL Servers (and their major versions-2005, 2008, 2008 R2, 2012) to be able to automatically refresh your CMS registrations on a regular basis as well as show some examples of how powerful the CMS can be for a small DBA team with a growing inventory.  Before getting started, you’ll want to set up an instance to be your central management server; see this URL for details: <http://msdn.microsoft.com/en-us/library/bb934126.aspx>.  **Background Information**  I work in an environment that started out with a relatively small number of SQL Servers (40) around 8 years ago. The environment has grown to well over 800 SQL servers today.  There are multiple ways to automate the management of this many systems (SSIS, .net custom apps, etc), but if someone wants a query executed on all non-production SQL Server 2008 systems, the quickest and easiest way to do so would be to run a query against a folder in the CMS that holds that specific subset of SQL Server Instances.  Some information about the environment that I work in:   * The instance inventory is assumed accurate (whose isn’t?)   + I use a separate SSIS package that connects to each instance in the inventory (version neutral) to populate the version data into the DBAV2 database * We break our instance management up into three categories: non-production, production non-critical, and mission critical * We use one CMS for managing all three categories of systems * We update the CMS registrations on a daily basis from our inventory (our categories are updated from our company CMDB daily to know what is prod versus non-prod)   **Breakdown of the Stored Procedures Used**  These MSDB stored procedures are undocumented, found by tracing SQL Server, so their behavior may change without notice, though I haven’t seen any changes from 2008 R2 to 2012.   * msdb.dbo.sysmanagement\_shared\_server\_groups (stores the CMS folders as seen in SSMS) * msdb.dbo.sp\_sysmanagement\_delete\_shared\_server\_group (deletes CMS folders) * msdb.dbo.sp\_sysmanagement\_add\_shared\_server\_group (adds CMS folder) * msdb.dbo.sp\_sysmanagement\_add\_shared\_registered\_server (adds instance to CMS folder)   There are two custom procedures built for using the MSDB procedures combined with my inventory:   * [dbo].[ap\_CMS\_Get\_Instance\_List]   + Based on parameters passed, this returns a list of instances to add to a specific folder (i.e. production non-critical > 2008)   + Parameters:     - @p\_isProd;  0=non-production, 1=prod non critical, 4=mission critical are valid values     - @p\_version; 2005, 2008, 2008 R2, 2012 are valid values     - @p\_isActive; 0=inactive, 1=active   + Pulls from the dbo.servers and dbo.instances tables * [dbo].[ap\_CMS\_Populate\_CMS\_folder]   + Used at the SQL Agent level to specify the high level folder name (prod non-critical, non-prod, mission critical) and then uses the [dbo].[ap\_CMS\_Get\_Instance\_List] to pull the list for each of the subfolders.   + Parameters:     - @p\_parent\_folder\_name; for this use, just looking at ‘non-production’, ‘production non-critical’ and ‘mission critical’ as values.   **Putting the Stored Procedures to Use**  Three scripts were provided to help with illustrating how simple, yet powerful, your CMS can be populated:   1. Script 01\_create\_inventory\_tables.sql will create a sample database for use as a demo, and will populate some dummy data into the inventory to help illustrate this 2. Script 02\_create\_procedures.sql will create the procedures that will be leverage to pull the instance list and create/populate the folder structure in the CMS 3. Script 03\_populate\_cms\_folders\_from\_inventory.sql will execute the [dbo].[ap\_CMS\_Populate\_CMS\_folder] for each of the three server categories that we wish to have.   To illustrate script 03\_populate\_cms\_folders\_from\_inventory.sql, this is what is happening at a high level:   1. EXEC [dbo].[ap\_CMS\_Populate\_CMS\_folder] 'Non-Production'    1. This procedure will drop and re-create the base folder named “Non-Production” and the following subfolders:       1. Create the subfolder for 2005 and populate the instances       2. Create the subfolder for 2008 and populate the instances       3. Create the subfolder for 2008 R2 and populate the instances       4. Create the subfolder for 2012 and populate the instances 2. EXEC [dbo].[ap\_CMS\_Populate\_CMS\_folder] 'Production'    1. This procedure will drop and re-create the base folder named “Non-Production” and the following subfolders:       1. Create the subfolder for 2005 and populate the instances       2. Create the subfolder for 2008 and populate the instances       3. Create the subfolder for 2008 R2 and populate the instances       4. Create the subfolder for 2012 and populate the instances 3. EXEC [dbo].[ap\_CMS\_Populate\_CMS\_folder] 'Mission Critical'    1. This procedure will drop and re-create the base folder named “Non-Production” and the following subfolders:       1. Create the subfolder for 2005 and populate the instances       2. Create the subfolder for 2008 and populate the instances       3. Create the subfolder for 2008 R2 and populate the instances       4. Create the subfolder for 2012 and populate the instances   The end result:  http://www.sqlservercentral.com/Images/22995.jpg  **Using your CMS Folders**  Now, we have our SQL Servers broken out by category and version, so if someone wants a query to be executed against all Mission Critical instances, you can right-click on the “Mission Critical” folder and open a new query window (this is by far the best use for CMS, in my opinion).  Be careful with some of the menu items, however, such as “Object Explorer”. Choosing Object Explorer will connect to each instance within the group, so use caution with selecting that if you have a lot of instances (a lot could be ~10) -- would you realistically want or need to connect to more than 10 in object explorer?  I would encourage you to go out and read into the added benefits of CMS from an admin perspective, but I wanted to illustrate how to run queries in the below examples.  http://www.sqlservercentral.com/Images/22996.jpg  Depending on how many servers are in the group, you should then see “Connected. (4/4)” at the bottom of the query window (assuming 4 SQL Servers are in the group, and all four are connected).  If 3 of the 4 connected, you would then see “Connected. (3/4)”.  http://www.sqlservercentral.com/Images/22997.jpg  **Examples of using T-SQL Queries against CMS Folders**  **To get data back from each instance in the selected group:** An example of a query execution is show below.  In this case, I want a quick result set of how many databases reside on each “Mission Critical” instance, regardless of version.  The query used was a query that would run on SQL 2005-2012 with no concerns, so the query executes on each instance in parallel and merges the result set for us.  The results are shown below:  http://www.sqlservercentral.com/Images/22998.jpg  To deploy a new database to each instance (of course, default data/log file locations would be used):  First, we can confirm there are no databases in existence with the name CMSExample1:  http://www.sqlservercentral.com/Images/22999.jpg  Now, I issue the create database statement:  http://www.sqlservercentral.com/Images/23000.jpg  And then to confirm that the database actually exists on each instance:  http://www.sqlservercentral.com/Images/23001.jpg  Then I want to clean up the test DB’s that I have created:  http://www.sqlservercentral.com/Images/23002.jpg  And to confirm:  http://www.sqlservercentral.com/Images/23003.jpg  Next steps/customizations   * Script 03\_populate\_cms\_folders\_from\_inventory.sql could be scheduled as a SQL Agent job to regularly refresh the CMS folder contents * Adding a section to [dbo].[ap\_CMS\_Populate\_CMS\_folder] to allow for SQL 2014 could be done (though I haven’t tested it) * You can manipulate the procedures to fit what you need (maybe you need a group per application or group per DBA that supports it)   **Conclusion**  I have shown that it is possible, with an existing inventory, to get a CMS folder structure set up for your mission critical, non-production, and production non-critical instances and easily be able to refresh it on a regular basis.  I showed some examples of how powerful multi-server queries can be at both creating objects on each instance, or by being able to quickly pull back data from multiple SQL Servers in your environment.  I hope this inspires you to “work smarter, not harder” as we all strive to be great DBA’s with less effort.  **References**  Set up a Central Management Server: <http://msdn.microsoft.com/en-us/library/bb934126.aspx>  Manage Your Environment with CMS: <http://www.sqlservercentral.com/articles/Central+Management+Server+%28CMS%29/88485/>  **Resources:**  [01\_create\_inventory\_tables.sql](http://www.sqlservercentral.com/Files/01_create_inventory_tables.sql/23004.sql) | [02\_create\_procedures.sql](http://www.sqlservercentral.com/Files/02_create_procedures.sql/23005.sql) | [03\_populate\_cms\_folders\_from\_inventory.sql](http://www.sqlservercentral.com/Files/03_populate_cms_folders_from_inventory.sql/23006.sql)  Thank this author by sharing:  Total article views: 768 | Views in the last 30 days: 768 |  |
|

**NOTE NOTE NOTE NOTE NOTE**

**THE ORIGINAL, UNMODIFIED SCRIPTS MENTIONED ABOVE CAN BE FOUND IN**

**AUTOMATE\_YOUR\_CENTRAL\_MANAGEMENT\_SERVER\_REGISTRATIONS.SQL**