Contents

[Introduction to EPM Framework 1](#_Toc223099003)

[1. Set up Database Objects To Store Policy Evaluation Results 2](#_Toc223099004)

[2. Set up Central Management Server 2](#_Toc223099005)

[3. Configure and Centralize Policies 4](#_Toc223099006)

[4. Design PowerShell Execution Strategy 5](#_Toc223099007)

[5. Configure PowerShell Execution 5](#_Toc223099008)

[6. Deploy Reports to SQL Server 2008 Reporting Services 7](#_Toc223099009)

# Introduction to EPM Framework

The Enterprise Policy Management Framework (EPM) is a solution to extend SQL Server 2008 Policy-Based Management to all versions of SQL Server in an enterprise, including SQL Server 2000 and SQL Server 2005. The EPM Framework will report the state of specified SQL Server instances against policies that define the defined intent, desired configuration, and deployment standards.

When the Enterprise Policy Management Framework (EPM) is implemented, policies will be evaluated against specified instances of SQL Server through PowerShell. This solution will require at least one instance of SQL Server 2008. The PowerShell script will run from this instance through a SQL Server Agent job or manually through the PowerShell interface. The PowerShell script will capture the policy evaluation output and insert the output to a SQL Server table. SQL Server 2008 Reporting Services reports will deliver information from the centralized table.

This solution requires the following components to be configured in your environment. All SQL Server 2008 requirements listed below may be executed from and managed on the same instance:

* SQL Server 2008 instance to store policies
* SQL Server 2008 instance to act as the Central Management Server
* SQL Server 2008 instance to execute the PowerShell script
* SQL Server management database and policy history table to archive policy evaluation results
* SQL Server 2008 Reporting Services to render and deliver policy history reports

Please refer to Microsoft documentation (<http://msdn.microsoft.com/en-us/library/cc645993.aspx>) to determine the appropriate editions to support central management server, policy evaluation, SQL Server Agent, and Reporting Services. All components are supported on SQL Server Enterprise and SQL Server Standard.

This document identifies the steps to configure the Enterprise Policy Management Framework objects in a SQL Server environment.

1. Set up Database Objects To Store Policy Evaluation Results

Policy evaluation results will be stored in a SQL Server table. This table should reside in a database designated for database management purposes. Many environments may choose to use a Management Data Warehouse created for Data Collectors, but this is not required. Sizing this table will depend on how many policies are evaluated, how many properties are evaluated in each policy, how many instances are evaluated, and how long the historical data will be maintained over time.

The script **EPM\_Create\_Database\_Table.sql** will create the database to store the policy history, the policy schema, the PolicyHistory table, and the EvaluationErrorHistory table. Open the SQLCMD script **EPM\_Create\_Database\_Table.sql**. Configure the variable ServerName with the name of the instance where you will store policy evaluation results. Note that the current version of the reports require the policy history database is stored on the same instance as the policies. Configure the variable ManagementDatabase with the name of the database where the table PolicyHistory will be created. See Figure 1.

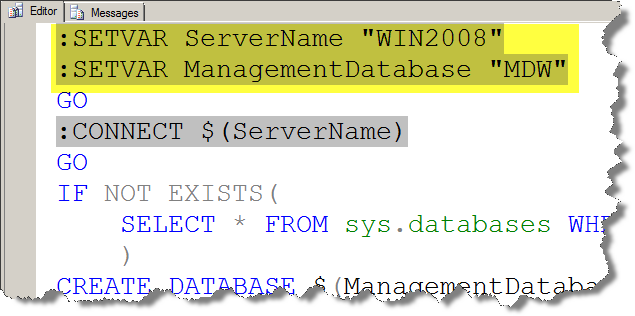


Figure 1

Execute the script. The script will perform the following steps:

1. If the database identified in the ManagementDatabase variable does not exist, the script will create the database.
2. The script will create a schema named policy if it does not exist.
3. The script will create the table PolicyHistory in the policy schema.
4. The script will create the table EvaluationErrorHistory in the policy schema.
5. Set up Central Management Server

The scripts provided in this solution assume that an instance of SQL Server 2008 is designated as a Central Management Server (CMS). The CMS will manage the logical groups of servers. The names of the logical groups are passed into variables in the Enterprise Policy Management Framework.

If an instance has not yet been designated, execute the following steps to register a Central Management Server:

1. Open SQL Server Management Studio. Select the View menu. Click Registered Servers.
2. In Registered Servers, expand Database Engine, right-click Central Management Servers, point to New, and then click Central Management Servers.
3. In the New Server Registration dialog box, register the instance of SQL Server that you want to become the Central Management Server.

After the instance has been designated as a CMS, manually register SQL Server instances into logical server group. See SQL Server Books Online details the manual steps to register instances into logical server groups: <http://msdn.microsoft.com/en-us/library/bb934126.aspx>.

Note: The current version of this solution will not support nested server groups. Nested server groups will be considered in future enhancements to this solution.

Database administrators managing large SQL Server environments with many SQL instances may prefer to automate the generation of server groups, and automate the registration of instances in the server groups. The following steps may be followed to automate the creation of server groups and registration of instances into the newly created groups. Many large SQL Server environments are currently managed via one or more lists of instances in a logical grouping. Minimally, these groups may include instances designated as production, test, development, etc. Environments may have additional group structures to denote departments (i.e. Finance, HR, Management), business function (i.e. ERP, CRM), severity levels, and/or regulatory requirements (i.e. PCI, HIPAA, SOX).

Open the SQLCMD script **CMS\_1\_Setup\_Management\_Server\_Tables.sql**. Configure the variable ServerName with the name of the instance which will store the CMS information. Configure the variable ManagementDatabase with the name of the database where the CMS tables will be created. The SQLCMD script **CMS\_1\_Setup\_Management\_Server\_Tables.sql** will create tables that may be populated with these logical groups of SQL Server instances. These tables provide a simple structure which may be populated with existing SQL Server groups from each organization. These tables will be created in the policy schema.

1. The SQLCMD script will create a policy schema if one does not exist.
2. The table policy.ServerGroup will store all logical server group names. These names may be Production, Development, Finance, SOX, etc.
3. The table policy.RegisteredServers will map instances to one or more groups.

After the script has completed, import existing server group data to the tables.

Open **CMS\_2\_Register\_Central\_Management\_Server\_From\_Management\_Tables.sql**. This SQLCMD script will execute a cursor based on the information stored in the policy.ServerGroup and policy.RegisteredServers tables. This script will create the CMS logical server groups as exist in the table policy.ServerGroup. The script will then register the instances as specified in policy.RegisteredServers. When the script completes, the Central Management Server will have a set of server groups and registered instances. is an example of a Central Management Server (individual server groups and registered instances will differ based on each environment).

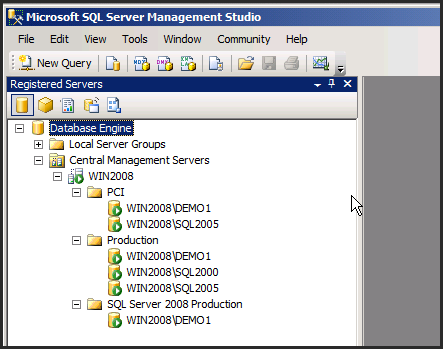


Figure 2

Larger environments which already store logical server groups in database tables may alter the SELECT statement in the SQLCMD script to loop through the existing SQL Server instance and group information.

1. Configure and Centralize Policies

All policies that will be evaluated using the EPM framework will be stored and managed from a single SQL Server 2008 instance. The PowerShell script will loop through these policies during execution. Often, the policies will be stored on the same SQL Server 2008 instance that acts as the Central Management Server. As noted earlier, the current version of the EPM Framework reports require will require that the policies are stored on the same instance as the database where the PolicyHistory table is stored.

Policies stored on the central server should be configured to improve scale of execution and minimize/eliminate false policy failures.

1. **Categorize policies**

The EPM framework PowerShell script will accept a parameter for policy category (example: Microsoft Best Practices: Performance). All policies should be placed in a category to enable a scalable solution for policy evaluation.

1. **Ensure all policies are configured with appropriate server restrictions and defined targets**

Policies may not be relevant for all versions and/or editions of SQL Server. For example, a policy which checks that a database is enabled for Transparent Data Encryption will not be relevant on SQL Server 2000 or SQL Server 2005, or SQL Server 2008 editions other than Enterprise. Furthermore, this policy may not be relevant for every database on an instance. Define server restrictions on polices to eliminate evaluation failures due to incompatible editions. Define targets on polices to eliminate false failures for databases which are not relevant to the policy.

1. Design PowerShell Execution Strategy

Determine the execution strategy for the EPM framework PowerShell script **EPM\_EnterpriseEvaluation.PS1** based on scale, execution time and security.

Security will be discussed in the section Configure PowerShell Execution.

This document assumes that the PowerShell script will be automated through SQL Server Agent or a third party scheduling agent. The execution strategy will execute a parallel or serial policy evaluation through multiple scheduled jobs and/or job steps.

* + PowerShell may be configured with multiple SQL Server Agent jobs for each policy category and/or each group of servers.
  + PowerShell may be configured with a single SQL Server Agent job and multiple steps for each policy category and/or each group of servers.

Administrators managing larger environments may want to implement policy evaluation through multiple parallel jobs that guarantee evaluation completion in the desired scope of time. A large environment that is challenged with a small maintenance window to execute the evaluation may design an execution strategy where the total number of concurrent jobs equals the total number of cores available for the instance.

1. Configure PowerShell Execution

The EPM\_EnterpriseEvaluation.PS1 PowerShell script requires variables are configured for the script to work with an enterprise environment.

*$HistoryServer:* Required.Centralized location of the SQL Server instance where the policy evaluation history database is located. This is the instance and database where the policy evaluation results are written.

*$HistoryDatabase:* Required. Name of the history database where the policy evaluation results are written.

*$SourceServer:* Required.SQL Server 2008 instance where polices are stored. This may be the same instance as the Central Management Server.

*$CMSServer:* Required.SQL Server 2008 instance which is defined as the Central Management Server. This instance will manage the logical groups of instances.

*$ConfigurationGroup:* Optional.Define the Central Management Server group to evaluate. If no value is provided, the PowerShell script will evaluate all servers registered in all of the CMS groups.

*$PolicyCategoryFilter:* Optional. Identifies which category of policies will be evaluated.

*$EvalMode:* Required. Specify the action to take during policy evaluation. Options are Check, Configure. Check will evaluate and report on the evaluation of the policy against the target. Configure will evaluate and report on the evaluation of the policy against the target, and reconfigure any settable and deterministic options that are not in compliance with the policies.

*$ResultDir:* Required. File location to write the policy evaluation results during the PowerShell execution. Results are written to this location temporarily.

After the variables are configured, the script will be placed in a scheduled job as a PowerShell type job step.

SQL Server Agent or a third party scheduling agent is required on the instance with SQL Server 2008 to automate the execution of the PowerShell script. SQL Server Agent on SQL Server 2005 does not support execution of PowerShell scripts. It is recommended that the EPM PowerShell script is executed from the instance defined as the Central Management Server, the instance where the policies are stored or the instance where the policy history is collected (these will often be the same instance). The following steps assume that SQL Server Agent is the scheduling agent for automating the PowerShell evaluation.

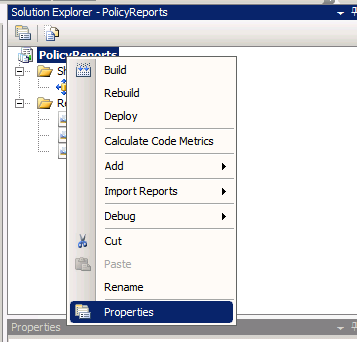
1. Create SQL Server Agent job(s) according to the design from step 4. As discussed I n step 4 above, larger environments will create multiple concurrent jobs for each category and/or CMS server group. Enterprise environments may prefer to create a single SQL Server Agent job with multiple steps for each category and/or CMS server group.
2. Create the job step(s) to execute PowerShell script.
   * 1. Configure variables. Each job and/or job step may specify a different policy category in the $PolicyCategoryFilter variable.These jobs/steps may also specify a different server group name in the $ConfigurationGroup variable.
     2. Define proxy account with appropriate level of access to instances and database objects which will be evaluated.

When polices are evaluated on demand through PowerShell, they will execute the policy evaluation in the context of the user issuing the evaluation. This account will require access to all instances and database objects the script will evaluate. The level of permissions will depend on what the policy is evaluating. This extends to the execution account used in a scheduling agent. In SQL Server 2008, the SQL Server Agent job step executes in the context of a specific user. This user may be a proxy account. The account that is specified in the SQL Server Agent step must have access to all objects on all instances that the PowerShell script will be evaluating.

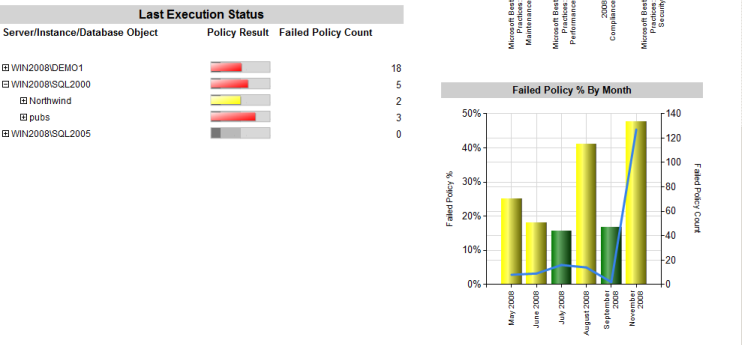
1. Deploy Reports to SQL Server 2008 Reporting Services

After the PowerShell script was successfully executed and the results of policy evaluation were collected in the “EvaluationResults” column of the “PolicyHistory” table, it is very convenient to visualize the data through Reporting Services reports. Steps below describe the process of report configuration and deployment.

1. Create views by executing the SQLCMD script **EPM\_CreateReportingViews.sql** script. The script will create four views. The view policy.v\_PolicyHistory will return the full historical dataset. The view policy.v\_PolicyHistory\_LastEvaluation returns the last execution status for each policy evaluation. These views will parse the XML data from EvaluationResults column.   
     
   Errors are stored in two locations. EvaluationErrorHistory table will store server connection errors and error history when the evaluation process cannot write to the PolicyHistory table. Errors are written to PolicyHistory when a policy fails to evaluate against the target. The view v\_EvaluationErrorHistory will combine the results from both error locations. The view v\_EvaluationErrorHistory\_LastEvaluation will return the last evaluation that resulted in an error.
2. Configure the Project properties with SSRS deployment options.
3. Open the PolicyReports project in Visual Studio 2008 and configure the PolicyDW.rds data source to point to the database on where the PolicyHistory table was created.
4. Right click on the PolicyReports project and select Properties. Set the appropriate TargetServerUrl, TargetReportFOlder and TargetDataSourceFolder properties of the “PolicyReports” project.



1. The visual indicators in the Last Execution Status table and the Failed Policy % By Month chart of the PolicyDashboard report will dynamically change color based on thresholds configured in hidden parameters of the report.



Before deploying the report, these parameters can be customized by selecting different warning threshold levels. To configure dashboard report parameters select the parameters @PolicyThresholdWarning (default is 0.5 – controls the Red “Warning” level) and @PolicyThresholdCaution (default is 0.17 – controls the Yellow “Caution” level).

1. Deploy the Reporting Services project. Right click on the PolicyReports project and select Deploy.