Windows PowerShell Get-Help on Cmdlet 'Invoke-PolicyEvaluation'

PS:\>Get-HELP Invoke-PolicyEvaluation -Full

NAME

Invoke-PolicyEvaluation

SYNOPSIS

Invokes one or more SQL Server policy-based management policy evaluations.

SYNTAX

Invoke-PolicyEvaluation [-Policy] <PSObject> [-AdHocPolicyEvaluationMode {Check | Configure | CheckSqlScriptAsProxy}] [-OutputXml] [-ProgressAction

<ActionPreference>] [-TargetExpression <String>] -TargetServerName <PSObject> [<CommonParameters>]

Invoke-PolicyEvaluation [-Policy] <PSObject> [-AdHocPolicyEvaluationMode {Check | Configure | CheckSqlScriptAsProxy}] [-OutputXml] [-ProgressAction

<ActionPreference>] -TargetObjects <PSObject[]> [<CommonParameters>]

DESCRIPTION

The Invoke-PolicyEvaluation cmdlet evaluates one or more policy-based management policies against a set of SQL Server objects named in the target set.

The policies specify the allowed values for various properties that are associated with SQL Server objects, such as establishing site standards for database names or

collations.

When this cmdlet runs in check mode, it reports whether the current properties of the objects in the target set comply with the rules in the policy definitions.

The objects in the target set are not reconfigured if their properties do not comply with the policies.

In configure mode, this cmdlet reconfigures any objects in the target set that do not comply with the policy definitions.

> `Module requirements: version 21+ on PowerShell 5.1; version 22+ on PowerShell 7.x.`

PARAMETERS

-AdHocPolicyEvaluationMode <AdHocPolicyEvaluationMode>

Specifies the adhoc policy evaluation mode. Valid values are:

- Check. Report the compliance status of the target set by using the credentials of your login account and without reconfiguring any objects.

- CheckSqlScriptAsProxy. Run a check report by using the ##MS_PolicyTSQLExecutionLogin## proxy account credentials.

- Configure. Reconfigure the target set objects that do not comply with the policies and report the resulting status. This cmdlet only reconfigures properties

that are settable and deterministic.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-OutputXml [<SwitchParameter>]

Indicates that this cmdlet produces its report in XML format using the Service Modeling Language Interchange Format (SML-IF) schema.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-Policy <PSObject>

Specifies one or more policies to evaluate.

Policies can be stored in an instance of the SQL Server database engine or as exported XML files.

For policies that are stored in an instance of the database engine, use a path that is based on the SQLSERVER:\SQLPolicy folder to specify the location of the

polices.

For policies that are stored as XML files, use a file system path to specify the location the policies.

This parameter can take a string that specifies the names of one or more policies to evaluate.

If only a file or policy name is specified in the string, this cmdlet uses the current path.

For policies that are stored in an instance of the database engine, use the policy name, such as "Database Status" or "SQLSERVER:\SQLPolicy\MyComputer\DEFAULT\Policies\Database Status." For policies that are exported as XML files, use the name of the file, such as "Database

Status.xml" or "C:\MyPolicyFolder\Database Status.xml."

This parameter can take a set of FileInfo objects, such as the output of Get-ChildItem run against a folder that contains

exported XML policies.

This parameter can also take a set of Policy objects, such as the output of Get-ChildItem run against a SQLSERVER:\SQLPolicy path.

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByValue)

Accept wildcard characters? false

-ProgressAction <ActionPreference>

Determines how PowerShell responds to progress updates generated by a script, cmdlet, or provider, such as the progress bars generated by the Write-Progress

cmdlet. The Write-Progress cmdlet creates progress bars that show a command's status.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-TargetExpression <String>

Specifies a query that returns the list of objects that define the target set.

The queries are specified as a string that has nodes which are separated by the '/' character.

Each node is in the format ObjectType[Filter].

ObjectType is one of the objects in the SQL Server Management Objects (SMO) object model, and Filter is an expression that filters for specific objects at that

node. The nodes must follow the hierarchy of the SMO objects. For example, the following query expression returns the AdventureWorks sample database:

[@Name='MyComputer']/Database[@Name='AdventureWorks']

If TargetExpression is specified, do not specify TargetObject.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-TargetObjects <PSObject[]>

Specifies the set of SQL Server objects against which the policy is evaluated. To connect to an instance of SQL Server analysis services, specify a

Microsoft.AnalysisServices.Server object for TargetObject.

If TargetObject is specified, do not specify TargetExpression.

Required? true

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-TargetServerName <PSObject>

Specifies the instance of the database engine that contains the target set.

You can specify a variable that contains a Microsoft.SqlServer.Management.Sfc.Sdk.SQLStoreConnection object.

You can also specify a string that complies with the formats that are used in the ConnectionString property of the

System.Data.SqlClient.SqlConnection class (v21

of the module) or the Microsoft.Data.SqlClient.SqlConnection class (v22+ of the module) in .Net.

These include strings such as those created by using either System.Data.SqlClient.SqlConnectionStringBuilder or the Microsoft.Data.SqlClient.SqlConnectionStringBuilder.

By default, this cmdlet connects by using Windows Authentication.

Required? true

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

INPUTS

System.Management.Automation.PSObject

OUTPUTS

NOTES

Example 1: Evaluate a policy on the default instance of the computer

PS C:\> Set-Location "C:\Program Files (x86)\Microsoft SQL Server\140\Tools\Policies\DatabaseEngine\1033"

PS C:\> Invoke-PolicyEvaluation -Policy "Trustworthy Database.xml" -TargetServer "MYCOMPUTER"

This command evaluate a policy on the default instance of the specified computer. The policy is read from an XML file and the connection is authenticated by using

Windows Authentication.

--------- Example 2: Evaluate policies from XML files ---------

PS C:\> Set-Location "C:\Program Files (x86)\Microsoft SQL Server\140\Tools\Policies\DatabaseEngine\1033"

PS C:\> Get-ChildItem "Database Status.xml", "Trustworthy Database.xml" | Invoke-PolicyEvaluation -TargetServer "MYCOMPUTER"

This command reads two policies from XML files in a folder, and then passes them to Invoke-PolicyEvaluation by using the pipeline operator.

Example 3: Evaluate policies and format the output according to the SMLIF schema

PS C:\> Set-Location "C:\Program Files (x86)\Microsoft SQL Server\140\Tools\Policies\DatabaseEngine\1033"

PS C:\> Invoke-PolicyEvaluation -Policy "Database Status.xml" -TargetServer "MYCOMPUTER" -OutputXML > C:\MyReportFolder\MyReport.xml

This command evaluates a policy and formats the output by using the Services Modeling Language Interchange Format (SML-IF) schema. The output is redirected to a file.

-------- Example 4: Evaluate a filtered set of policies --------

PS C:\> Set-Location "SQLSERVER:\SQLPolicy\MYCOMPUTER\DEFAULT\Policies"

PS C:\> Get-ChildItem | Where-Object { $_.PolicyCategory -eq "Microsoft Best Practices: Maintenance" } | Invoke-PolicyEvaluation -TargetServer 'MYCOMPUTER'

The first command sets the current path to a SQL Server policy store.

The second command uses Get-ChildItem to read all of the polices and then uses Where-Object to filter the list for the policies that have their PolicyCategory

property set to "Microsoft Best Practices: Maintenance".

The output is sent to Invoke-PolicyEvaluation by using the pipeline operator.

Example 5: Evaluate policies from XML files by using a SqlStoreConnection object

PS C:\> Set-Location "C:\Program Files (x86)\Microsoft SQL Server\140\Tools\Policies\DatabaseEngine\1033"

PS C:\> $Connection = New-Object Microsoft.SqlServer.Management.Sdk.Sfc.SqlStoreConnection("server='MYCOMPUTER';Trusted_Connection=True")

PS C:\> Invoke-PolicyEvaluation -Policy "Database Status.xml" -TargetServer $Connection

The first command sets the current location to a local folder that contains policy evaulations in XML files.

The second command uses New-Object to create a SqlStoreConnection object.

The third command evaluates policy from an XML file against the server defined by the SqlStoreConnection object.

- Example 6: Evaluate policy using a manually loaded assembly -

PS C:\> Set-Location "C:\Program Files (x86)\Microsoft SQL Server\140\ tools\Policies\analysisservices\1033"

PS C:\> [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.AnalysisServices")

PS C:\> $SSASsvr = New-Object Microsoft.AnalysisServices.Server

PS C:\> $SSASsvr.Connect("Data Source=localhost")

PS C:\> Invoke-PolicyEvaluation "Surface Area Configuration for Analysis Services Features.xml" -TargetObject $SSASsvr

The first command sets the current folder location.

The second command loads an instance of the SQL Server Analysis Services assembly.

The third command creates a Microsoft.AnalysisServices object.

The fourth command uses the new AnalysisServices object to open a connection to the default server instance on the local computer.

The fifth command evaluates the Analysis Services surface area configuration policy.

-------- Example 7: Evaluate a filterd set of policies --------

PS C:\> Set-Location "C:\Program Files (x86)\Microsoft SQL Server\120\Tools\Policies\DatabaseEngine\1033"

PS C:\> Invoke-PolicyEvaluation "Database Status.xml" -TargetServer "MYCOMPUTER" -TargetExpression "Server[@Name='MYCOMPUTER']/Database[@Name='AdventureWorks2014']"

This command uses the TargetExpression parameter to specify a query expression that filters the database status policy be evaluated against the AdventureWorks2014

sample database and performs the evaluation.

Example 8: Evaluate the reporting services surface area configuration policy

PS C:\> Set-Location "C:\Program Files (x86)\Microsoft SQL Server\120\Tools\Policies\ReportingServices\1033"

PS C:\> [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SqlServer.Dmf.Adapters")

PS C:\> $SSRSsvr = New-Object Microsoft.SqlServer.Management.Adapters.RSContainer('MyComputer')

PS C:\> Invoke-PolicyEvaluation -Policy "Surface Area Configuration for Reporting Services 2008 Features.xml" -TargetObject $SSRSsvr

This command loads the SQL Server Reporting Services assembly, creates a connection to the default server instance on the local computer, and runs the Reporting

Services surface area configuration policy.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/sqlserver/invoke-policyevaluation

SQLServer_Cmdlets