SQL Sentry PowerShell Module

Last Modified on 25 May 2021

Getting Started

PowerShell Requirements

The following versions of **Windows PowerShell** are compatible with the SQL Sentry **PowerShell** module:

- 3.0
- 4.0
- 5.0
- 5.1

> **Note:** **PowerShell Core** is not supported.

The SQL Sentry installation package includes a **PowerShell** module that can be used to manage your SQL Sentry environment through **PowerShell**. This topic includes a walkthrough of that functionality.

Importing the module

You'll find the PowerShell module in the SQL Sentry program directory (which varies depending on the installed):

Versions 2021.8 or later:

> C:\Program Files\SolarWinds SQL Sentry\\Intercerve.SQLSentry.Powershell.psd1

Versions prior to 2021.8:

> C:\Program Files\SentryOne\\Intercerve.SQLSentry.Powershell.psd1

Import the SQL Sentry **PowerShell** module by using the **Import-Module** command.

Import-Module command example for version 2021.x:

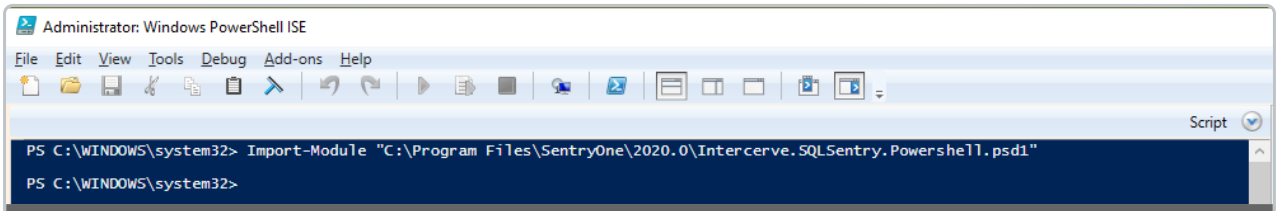
```
Import-Module "C:\Program Files\SolarWinds SQL Sentry\2021.0\Intercerve.SQLSentry.Powershell.psd1"
```

Import-Module command example for version *19.x*:

```
Import-Module "C:\Program Files\SentryOne\19.0\Intercerve.SQLSentry.Powershell.psd1"
```

Import-Module command example for version *20.x*:

```
Import-Module "C:\Program Files\SentryOne\20.0\Intercerve.SQLSentry.Powershell.psd1"
```



Import-Module Example with version 2020.0

Note: When using the Enhanced Platform Installer (EPI), the module location is as follows (if the machine has both a client and monitoring service installed, the module will appear in both locations):

Monitoring Service

On a machine with the monitoring service installed:

C:\ProgramData\SentryOne\monitoringservice\bin\Intercerve.SQLSentry.Powershell.psd1

Client

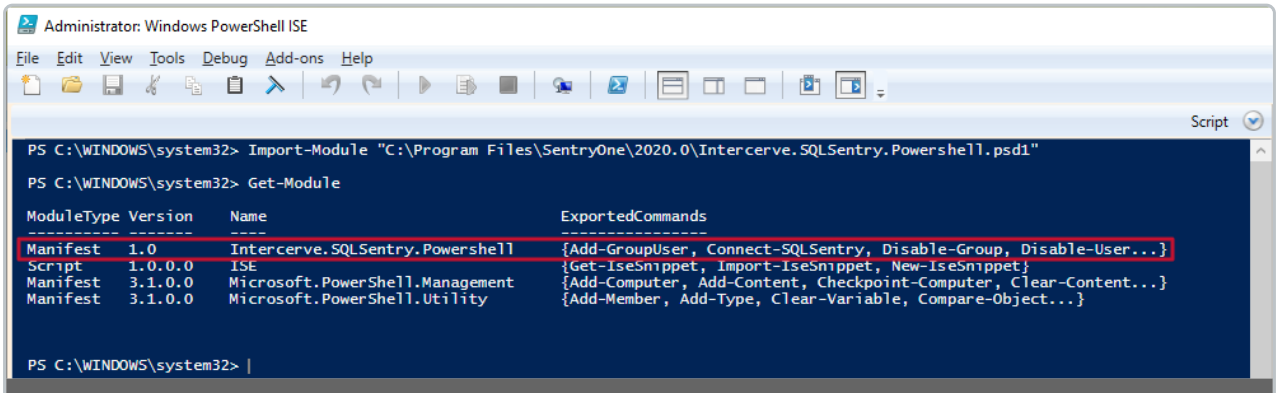
On a machine with the client installed:

C:\Users\\Documents\sentryone\client\\\bin\Intercerve.SQLSentry.Powershell.psd1

For example:

C:\Users\mconnors\Documents\sentryone\client\QA-SRV-1\SentryOne\bin\Intercerve.SQLSentry.Powershell.psd1

Verify that the module imported correctly by running the **Get-Module** command. You should see a listing for **Intercerve.SentryOne.Powershell**.



Get-Module Example showing the Intercerve.SQLSentry.PowerShell

Additional Information: For more information about **Get-Module** commands, see Microsoft Get Module Commands.

Available Commands

See the Command Examples section for more information on using available commands.

Command	Description
Connect-SQLSentry	Allows you to connect to a specific SQL Sentry installation that's required before any other actions are performed. This command is useful for navigating between repositories, and is used in environments with more than one SQL Sentry database. **Available parameters** • DatabaseName • Login • Password • ServerName • UseIntegratedSecurity
Disable-User	Disable a SQL Sentry user account.
Disconnect-SQLSentry	Allows you to disconnect from a specific SQL Sentry installation. Useful for environments with more than one SQL Sentry database.
Enable-User	Enable a SQL Sentry user account.
Get-Computer	Allows you to view computers in your environment. **Available Parameters** • ComputerType ◦ {*Windows* | *VMwareHost* | *ApsAppliance* | *SqlDataWarehouse* | *AzureSqlDatabase* | *AzureElasticPool*} • ID • Name **Information returned** • Name • HostName • DomainName • ObjectID • ID • ComputerType • AccessLevel
	Allows you to view connections in your environment. **Available parameters** • ConnectionType ◦ {*SqlServer* | *SqlServerAnalysisServices* | *RDSSqlServer*} • ID

Command	Description
Get-Connection	• Name • ObjectID **Information returned** • Name • ServerName • InstanceName • ObjectID • ID • ConnectionType • WatchedBy
Get-Group	Allows you to view groups in your environment. **Available Parameters** • Name • Username **Information returned** • Name • Description • IsEnabled • ObjectID • ID
Get-Site	Allows you to view sites in your environment. **Note:** This includes both sites and groups in the return. **Available Parameters** • ID • Name
Get-SQLSentryConfiguration	Allows you to view basic information about your SQL Sentry configuration. **Information returned** • SQLServer • Database • UseIntegratedSecurity • DatabaseLogin • DatabaseVersion • ApplicationDatabaseVersion • ApplicationPath • ApplicationVersion • ServerTime • ServerTimeUtc
	Allows you to view information about users in your environment. **Note:** See examples in the code section below for piping with Add-GroupUser and Remove-GroupUser to manager group memberships for users. **Available Parameters** • FirstName • Name

Command	Description
Get-User	• EmailAddress • ID • Group **Information returned** • FirstName • LastName • EmailAddress • PagerAddress • Description • IsEnabled • ObjectID • ID
Invoke-UnwatchConnection	Allows you to stop watching a connection with SQL Sentry. **Available Parameters** • Connection
Invoke-WatchComputer	Allows you to watch a computer with SQL Sentry. **Available Parameters** • Computer • LicenseMode
Invoke-WatchConnection	Allows you to watch a connection with SQL Sentry. **Available Parameters** • Connection
Register-Computer	Allows you to register a computer to be watched with SQL Sentry. **Available Parameters** • ComputerType • Name • AccessLevel • DatabaseName • Login • Password • Port • TargetSite • UseIntegratedSecurity • AllowAzureRemoteObjectInstallation **Note: AllowAzureRemoteObjectInstallation** is only valid for Azure SQL Database targets. **Additional Information:** For more information about adding servers, see the Automating adding servers to SentryOne article.
	Allows you to register a connection to be watched with SQL Sentry.

Command	Available Parameters Description
Register-Connection	• ConnectionType • Name • Login • Password • Port • TargetSite • UseIntegratedSecurity
Register-Group	Adds a new user group.
Register-User	Adds a new SQL Sentry user.
Unregister-Group	Removes an existing user group.
Unregister-User	Removes an existing SQL Sentry user.

Advisory Conditions & Configurations

Note: The advisory condition and configuration commands in the following tables are only available in SQL Sentry versions 2021.1.13 or later.

Advisory Conditions

The advisory condition commands use the same logic that is found in the individual import and export features on the SQL Sentry client menu, but they allow for bulk importing and exporting of **.condition** files.

Command	Description
Export-AdvisoryCondition	Returns one or more advisory condition file items. **Note:** You can filter advisory conditions by using `ConvertFrom-Json` and then operating on any advisory condition object properties. **Required** • Must use `Out-File -FilePath` to write to file system. ◦ `Import-AdvisoryCondition` command can only use files. • Must use `foreach` to write out multiple files. **Parameters** • Name ◦ Advisory condition name search pattern which can include wildcards (%).
	Imports one or more advisory conditions. **Parameters** • Path

Command	Description
Import-AdvisoryCondition	◦ File or directory name. • SearchPattern ◦ File name pattern to use with directory can which can include wildcards (%). ▪ Default is **.condition*
Remove-AdvisoryCondition	Deletes one or more advisory conditions. **Parameters** • Name ◦ Advisory condition name search pattern which can include wildcards (%).

Configurations

The configuration file is stored in a JSON document and may be edited as needed.

The configuration commands include the following in the import/export:

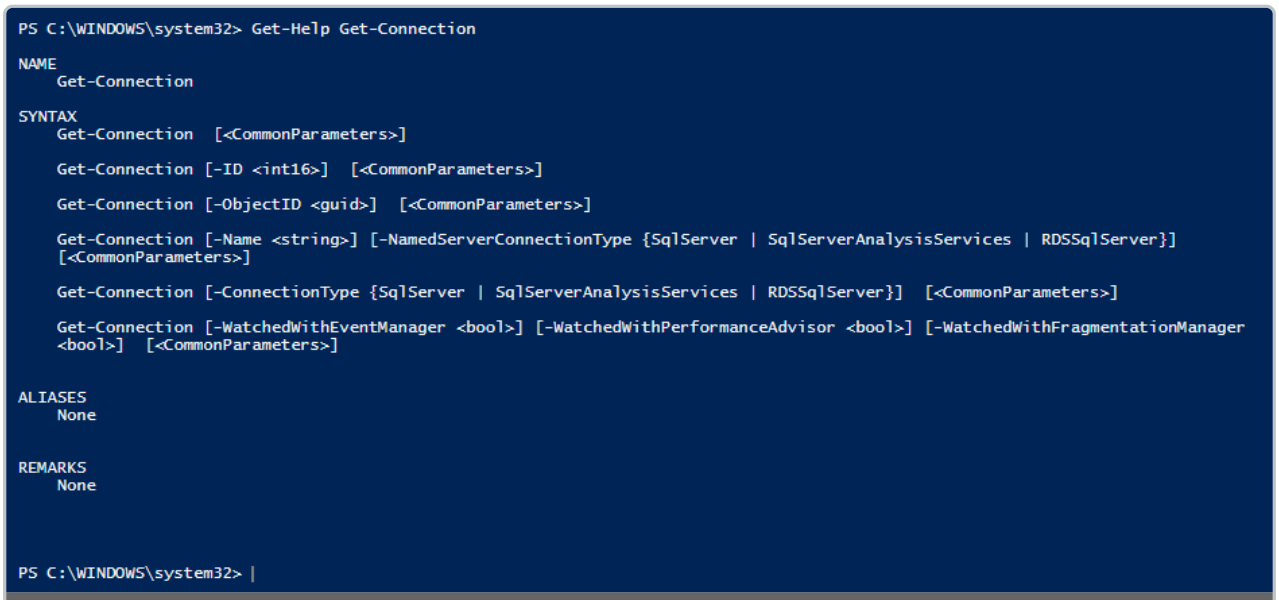
- Advisory Conditions
- Conditions/Actions
  - Rulesets, Windows, Filters, etc.
- Users and Groups (assigned)
- All Global (All Targets) Settings

Command	Description
Export-EnvironmentConfiguration	Exports a SQL Sentry system configuration in JSON format. The export may be written to a variable or file. **Switches** • ExcludeAdvisoryConditions • ExcludeContacts • ExcludeConditionActions • ExcludeGlobalSettings
Import-EnvironmentConfiguration	Imports and applies a SQL Sentry system configuration from a variable or file in a JSON format. **Parameters** • Config ◦ Variable containing the JSON configuration. • Path ◦ Path to the JSON configuration file. **Switches** • ExcludeAdvisoryConditions • ExcludeContacts • ExcludeConditionActions • ExcludeGlobalSettings

Get-Help

Use **Get-Help** followed by a command to get information about the available commands in the table above. For example, to learn more about **Get-Connection**:

```
Get-Help Get-Connection
```



Get-Help Get-Connection Example

Command Examples

You can get a file of all these command examples from GitHub in the SentryOne PowerShell Module Commands.ps1 file.

Import the SQL Sentry PowerShell Module

```
<# Import the SQL Sentry PowerShell Module #>
Import-Module "C:\Program Files\SentryOne\2020.0\Intercerve.SQLSentry.Powershell.psd1"
```

Connect to SQL Sentry Installation

```
<# Connect to a specific SQL Sentry Installation #>
Connect-SQLSentry -ServerName server.domain.com -DatabaseName SQLSentry
```

Get SQL Sentry Installation Information

```
<# Get Information about your SQL Sentry Installation #>
Get-SQLSentryConfiguration
```

Get SQL Sentry Information for Sites

```
<# Get Information about the Sites in your SQL Sentry Installation -use parameters to find informat
ion for a specific site #>
Get-Site
```

Get SQL Sentry Information for Instances

```
<# Get Information about the Connections (Instances) in your SQL Sentry Installation -use parameter
s to find information for a specific connection #>
Get-Connection
```

Get SQL Sentry Information for Targets

```
<# Get Information about the Computers (Targets) in your SQL Sentry Installation -use parameters to
find information for a specific connection #>
Get-Computer
```

Get SQL Sentry Information for Specific Target

```
<# Get Information about the Connections in your SQL Sentry Installation -use parameters to find in
formation for a specific connection #>
Get-Connection
```

Register a Target

```
<# Register Computers (Targets), so that they can be watched in your environment #>
Register-Computer -ComputerType Windows -Name server.domain.com -AccessLevel Full Register-Computer
-ComputerType Windows - Name server.domain.com -AccessLevel Limited
```

Register a Target (non Windows)

```
<# Register a Target that cannot utilize Windows Authentication (e.g., Azure SQL Database) #>
Register-Computer -ComputerType AzureSqlDatabase -Name example.database.windows.net -DatabaseName
dbName -Login username -Password password -AccessLevel Full -UseIntegratedSecurity 0
```

Register Instances

```
<# Register Connections (Instances), so that they can be watched in your environment #>
Register-Connection -ConnectionType SqlServerAnalysisServices -Name server.domain.com Register-Conn
ection -ConnectionType SqlServer -Name server.domain.com
```

Watch a Windows Target

```
<# Watch Windows Computer (Target) with SQL Sentry | -Pipe in the Computer #>
Get-Computer -Name server.domain.com -NamedServerComputerType Windows | Invoke-WatchComputer
```

Watch a Hyper-V Host Target

```
<# Watch Hyper-V Host (Target) with SQL Sentry (core-based licensing) | -Pipe in the Computer #>
Get-Computer -Name server.domain.com -NamedServerComputerType Windows | Invoke-WatchComputer -Licen
seMode CoreBased
```

Watch a SQL Server Target

```
<# Watch SQLServer Connection (Instance) with SQL Sentry | -Pipe in the Connection #>
Get-Connection -Name server.domain.com -NamedServerConnectionType SqlServer | Invoke-WatchConnectio
n
```

Watch a SSAS Target

```
<# Watch SSAS Connection (Instance) with SQL Sentry | -Pipe in the Connection #>
Get-Connection -Name server.domain.com -NamedServerConnectionType SqlServerAnalysisServices | Invok
e-WatchConnection
```

Unwatch Target

```
<# Unwatch Windows computer (Target) #>
Get-Computer -Name server.domain.com -NamedServerComputerType Windows | Invoke-UnwatchComputer
```

Unwatch SSAS Connection

```
<# Unwatch SSAS connection #>
Get-Connection -Name server.domain.com -NamedServerConnectionType SqlServerAnalysisServices | Invok
e-UnwatchConnection
```

Unwatch SQL Server Connection

```
<# Unwatch SQLServer connection #>
Get-Connection -Name server.domain.com -NamedServerConnectionType SqlServer| Invoke-UnwatchConnecti
on
```

User cmdlets

```
<# User cmdlets #>

<# Add a user #>
Register-User -FirstName Test -LastName User -Email tuser@test.net -PagerAddress tuser@testPager.ne
t -Description Tester -Login domain\username

<# View a user by first name #>
Get-User -FirstName Test

<# View a user by name #>
Get-User -Name "Test User"

<# Disable a user by name #>
Disable-User -Name "Test User"

<# Enable a user by name #>
Enable-User -Name "Test User"

<# Remove a user by name #>
Unregister-User -Name "Test User"
```

Group cmdlets

```
<# Group cmdlets #>

<# Create a new group #>
Register-Group -Name "Test Group" -Description "A Group" -Login Domain\TestGroup

<# View group information #>
Get-Group -Name "Test Group"

<# Disable a group #>
Disable-Group -Name "Test Group"

<# Enable a group #>
Enable-Group -Name "Test Group"

<# Remove a group #>
Unregister-Group -Name "Test Group"
```

User and Group cmdlets

```
<# User = and - to the Group cmdlets #>

<# Add a user to a group #>
Get-User -Name "Test User" | Add-GroupUser -GroupName "Test Group"

<# Remove a user from a group #>
Get-User -Name "Test User" | Remove-GroupUser -GroupName "Test Group"
```

> ⚠ **Important:** For performance considerations, when using scripts to bulk add targets, we recommend limiting the text file list to batches of 50-100 targets at a time.

PowerShell Sample Scripts

The following code samples are available in the sentryone-samples / powershell repository on GitHub.

- Watch all targets listed in text file.ps1
- Watch all targets listed in text file (Azure).ps1

Export/Import/Remove Advisory Conditions

```
$exportdirectory = "C:\Users\\Documents\conditions"
 $count = 0
foreach ($ac in Export-AdvisoryCondition)
{
    $acj = $ac | ConvertFrom-Json
    # Use .* for all ACs; separate multiple tags with vertical pipes (|).
    $tags = '.*'
    if ($acj.Tags -match $tags)
    {
        # Replace invalid file name chars
        $filename = ($acj.Name -replace '[/<>]', '_') + ".condition"
        $ac | Out-File -FilePath "$exportdirectory\$filename"
        Write-Host "Exported: $filename"
        $count++
    }
}
Write-Host "$count conditions exported."

Import-AdvisoryCondition $exportdirectory # import all ACs
Import-AdvisoryCondition "$exportdirectory\High Compiles.condition"
Import-AdvisoryCondition -Path $exportdirectory -SearchPattern "Service Broker*"

Remove-AdvisoryCondition -Name "High Compiles"
Remove-AdvisoryCondition -Name "Service Broker%"
Remove-AdvisoryCondition -Name "%" # will not cascade delete dependent ACs
```

Export/Import Configurations

Using a configuration file

```
# Using config file
$exportdirectory = "C:\Users\\Documents\conditions"

Connect-SQLSentry -ServerName "server-name-us" -DatabaseName "SQLSentry-US"
Export-EnvironmentConfiguration -ExcludeAdvisoryConditions | Out-File -FilePath "$exportdirectory\e
nv-config.json"
Disconnect-SQLSentry

Connect-SQLSentry -ServerName "server-name-uk" -DatabaseName "SQLSentry-UK"
Import-EnvironmentConfiguration -Path "$exportdirectory\env-config.json"
Disconnect-SQLSentry
```

Using a variable

```
# Using variable
Connect-SQLSentry -ServerName "server-name-us" -DatabaseName "SQLSentry-US"
$s1config = Export-EnvironmentConfiguration
Disconnect-SQLSentry

Connect-SQLSentry -ServerName "server-name-uk" -DatabaseName "SQLSentry-UK"
Import-EnvironmentConfiguration -Config $s1config
Disconnect-SQLSentry
```