SQL Sentry Advisory Conditions on GitHub

Last Modified on 18 May 2021

Introduction

Additional SQL Sentry Advisory Conditions are available through GitHub. In **sentryone / advisory-conditions**, there is a folder called **s1-team-submitted** that contains conditions outside of the standard Advisory Conditions Pack. These conditions have been created by SQL Sentry team members (often the sales engineering and support teams) and used across many of our customer installations. They are now available on GitHub to make it easier to share them with everyone.

Note:

- These conditions may be less-universally applicable than the ones in the standard download pack and you're encouraged to pick and choose the ones applicable to your needs.
- These advisory conditions are provided as samples to get you started.
  - They may require additional setup or configuration to meet your needs (setting variables and changing predicates in T-SQL, how often they evaluate, color highlighting, etc.). Please read the descriptions for guidance.
- The conditions listed in the **sentryone / advisory-conditions / download-pack** folder are already part of your installation. Please ignore these for now as they are part of the standard Advisory Conditions Pack.

Downloading

Download: Go to https://github.com/sentryone/advisory-conditions

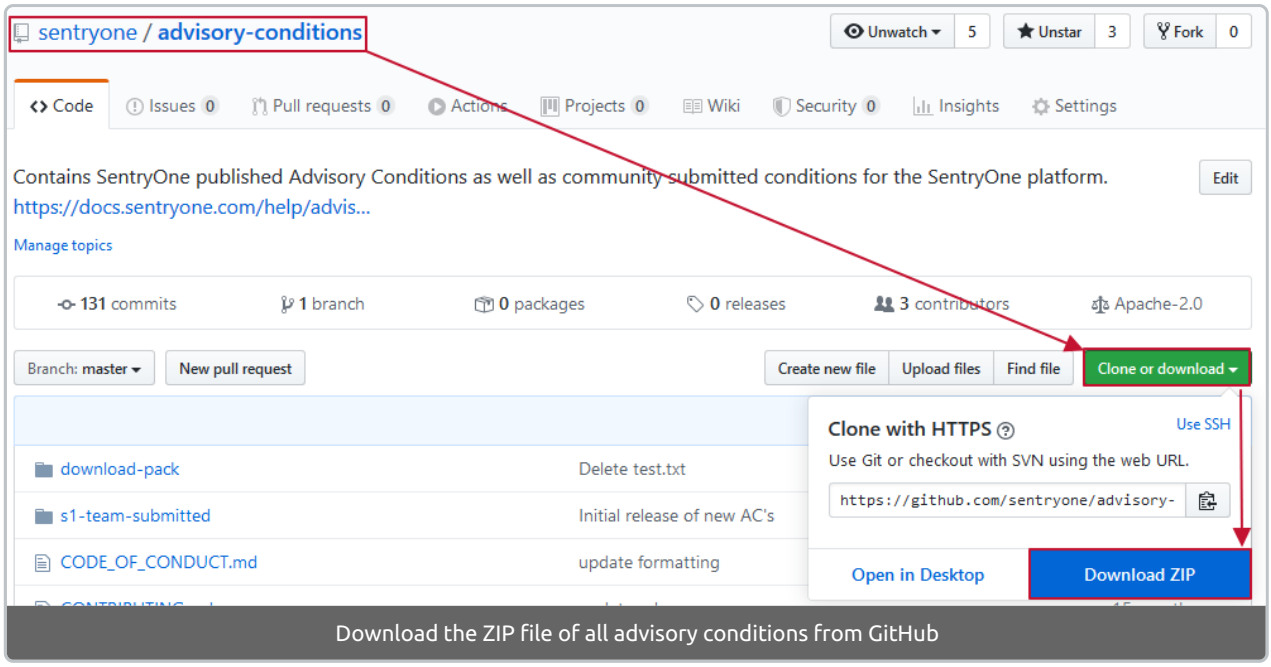
Downloading conditions from GitHub

Download zip file of all conditions (recommended)

This method will download all the advisory conditions available from the repository and allow you to import the conditions you want to configure for your SQL Sentry environment.

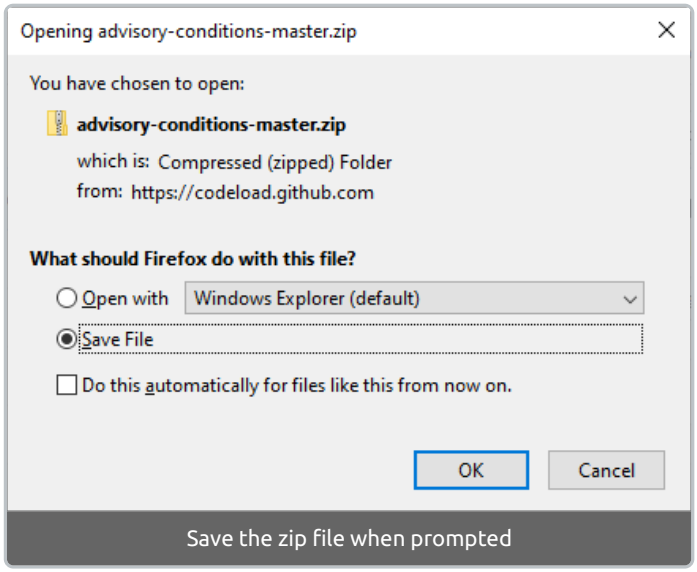
Note: Individual advisory conditions are tiny text files, so the entire zip file is currently under 1 MB.

From **sentryone / advisory conditions**, select the **Clone or download** button, then select **Download ZIP**.



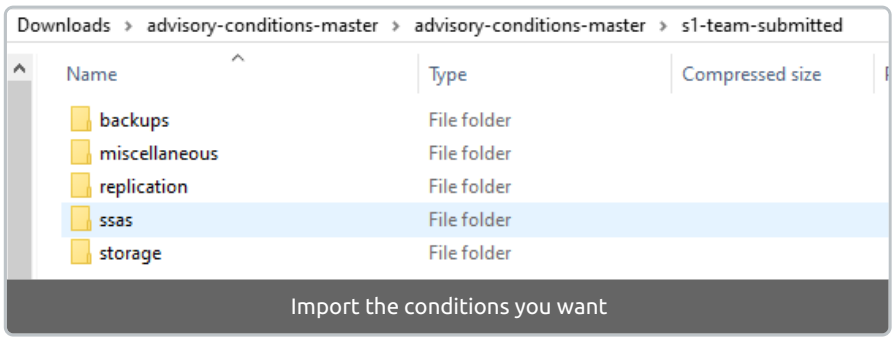
Download the ZIP file of all advisory conditions from GitHub

You'll be prompted to save **advisory-conditions-master.zip**:



Save the zip file when prompted

Once you have the files, you import them into SQL Sentry through the client. See the Advisory Conditions article for instructions on accessing **Import** from the context menu.

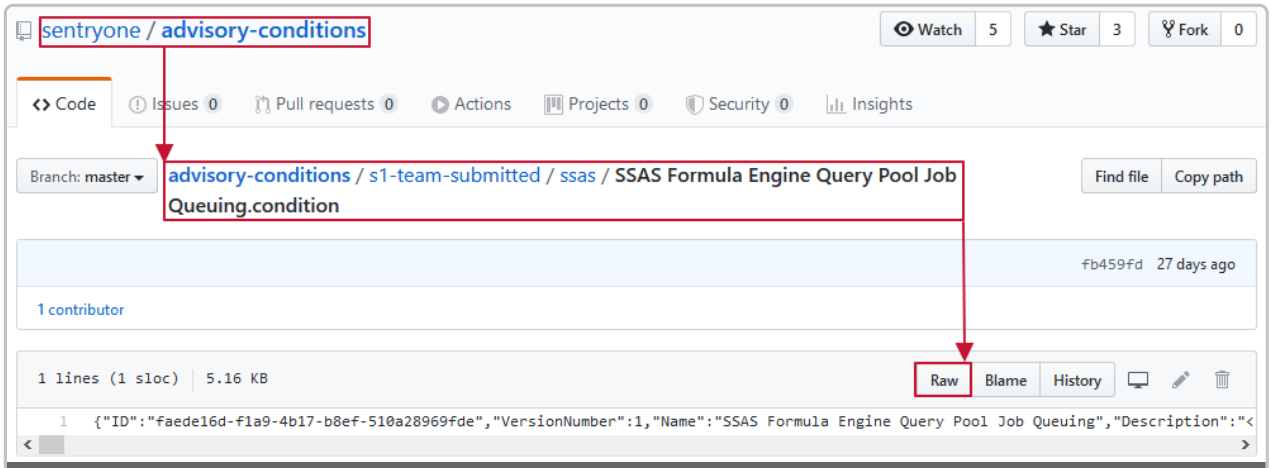


Import the conditions you want

Import the individual conditions that you want to configure for your environment. Don't forget to set up the appropriate alerts and actions.

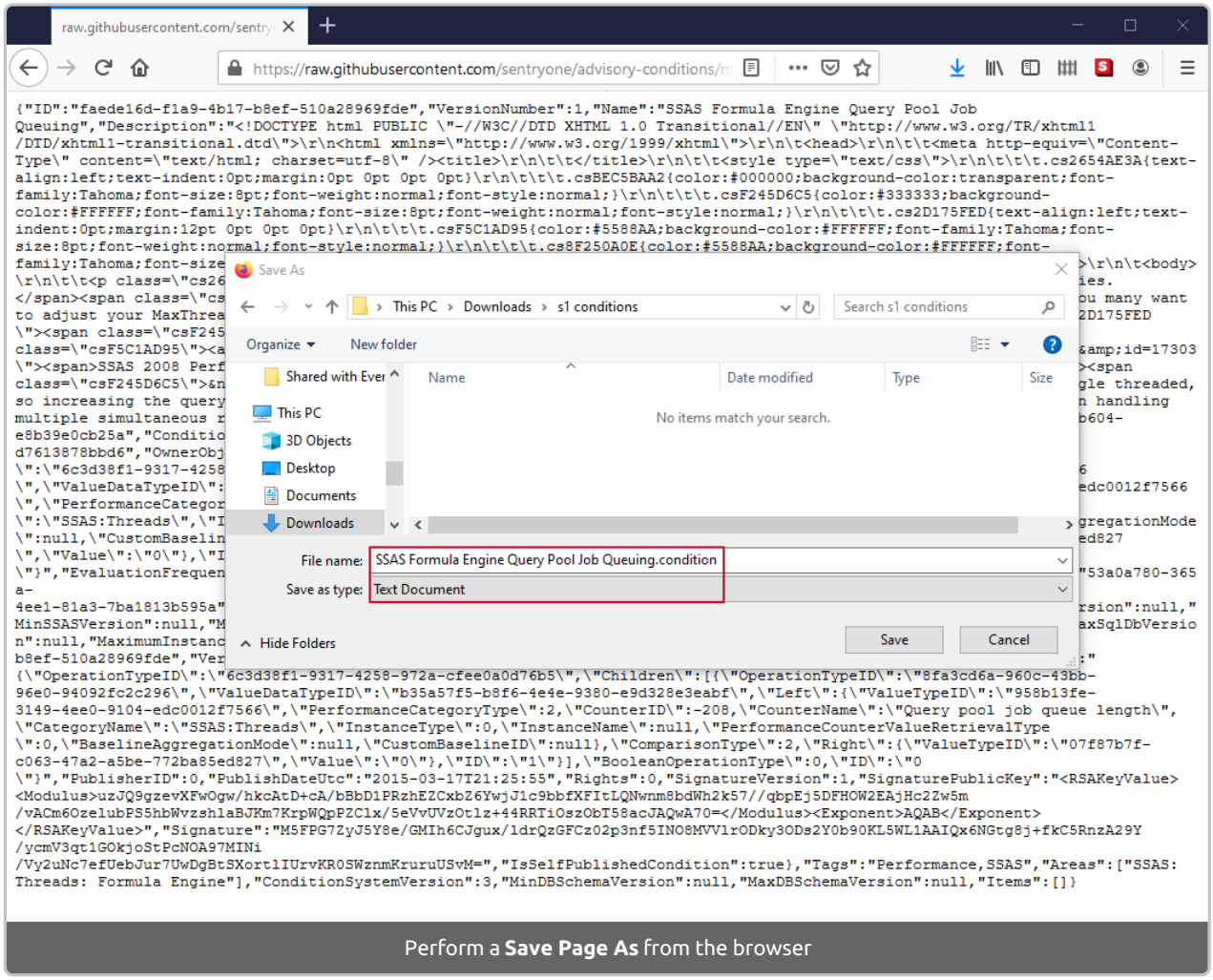
Download an individual condition

Navigate to an individual condition that you want to download and select the **Raw** button:



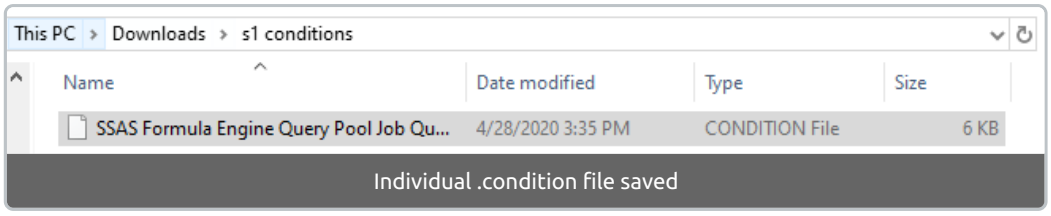
Select the **Raw** button from the individual condition

On the **Raw** page, you should be able to do a **Save Page As** command in your browser and save it as a **text file** that ends in **.condition**:



Perform a **Save Page As** from the browser

Now you'll have the individual condition to import through the SQL Sentry client:



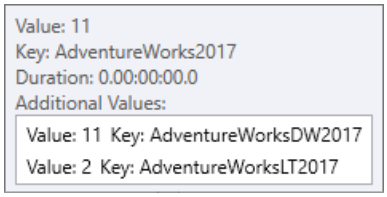
Available Conditions

s1-team-submitted / auditing

Additional Information: See the Using SQL Sentry to Alert on SQL Server Database Object Changes blog post for instructions on how to use/customize these **auditing** conditions and apply the **Execute SQL** actions.

Results format:

These advisory conditions return a list of databases (**Key**) and the number of objects (**Value**) meeting the condition criteria. The default results look like this:



In the example above, the *AdventureWorksDW2017* and *AdventureWorksLT2017* databases each have objects (*11* and *2*, respectively) that meet the condition criteria. To get the list of objects (e.g. stored procedures, tables) you must run an additional query (manually) or set an action to run the query and email the full list of results to you (automatically).

Stored Procedures - Created

This condition evaluates to *True* when it finds databases with stored procedures that have been created with the past hour.

The results returned are number of procedures in a database that meet the condition (**Value**) and the name of the database that has the created procedures (**Key**).

If the condition evaluates to *True*, run the following query against the database(s) in the result set to get a list of all procedures created **today**:

```
SELECT schema_name(schema_id), [name], create_date
FROM sys.procedures
WHERE create_date >= (SELECT CONVERT(DATE,GETDATE()))
ORDER BY create_date DESC;
```

An **Execute SQL** action can also be used to automatically email a list to you. See the sentryone-samples repository on GitHub for a sample you can use.

Stored Procedures - Modified

This condition evaluates to *True* when it finds databases with stored procedures that have been modified with the past hour.

The results returned are number of procedures in a database that meet the condition (**Value**) and the name of the database that has the modified procedures (**Key**).

If the condition evaluates to *True*, run the following query against the database(s) in the result set to get a list of all procedures modified **today**:

```
SELECT schema_name(schema_id), [name], modify_date
FROM sys.procedures
WHERE modify_date >= (SELECT CONVERT(DATE,GETDATE()))
AND modify_date != create_date
ORDER BY modify_date DESC;
```

An **Execute SQL** action can also be used to automatically email a list to you. See the sentryone-samples repository on GitHub for a sample you can use.

Tables - Created

This condition evaluates to *True* when it finds databases with tables that have been created with the past hour.

The results returned are number of tables in a database that meet the condition (**Value**) and the name of the database that has the created tables (**Key**).

If the condition evaluates to *True*, run the following query against the database(s) in the result set to get a list of all tables created **today**:

```
SELECT schema_name(schema_id), [name], create_date
FROM sys.tables
WHERE create_date >= (SELECT CONVERT(DATE,GETDATE()))
ORDER BY create_date DESC;
```

An **Execute SQL** action can also be used to automatically email a list to you. See the sentryone-samples repository on GitHub for a sample you can use.

> **Note:** If you have the SQL Sentry Scalability Pack, it's normal to see a table named **[Partitioning].[PartitionTracking_backup]** created in the SQL Sentry database on a regular basis. You may want to filter out this table name in the queries.

Tables - Modified

This condition evaluates to *True* when it finds databases with tables that have been modified with the past hour.

The results returned are number of tables in a database that meet the condition (**Value**) and the name of the database that has the modified tables (**Key**).

If the condition evaluates to *True*, run the following query against the database(s) in the result set to get a list of all tables modified **today**:

```
SELECT schema_name(schema_id), [name], modify_date, create_date
FROM sys.tables
WHERE modify_date >= (SELECT CONVERT(DATE,GETDATE()))
AND modify_date != create_date
ORDER by modify_date DESC;
```

An **Execute SQL** action can also be used to automatically email a list to you. See the sentryone-samples repository on GitHub for a sample you can use.

Note:

- The modified date is updated when there are changes to indexes and partitions associated with the table (i.e. adding a new partition, deleting an index, etc.).
  - If you have the SQL Sentry Scalability Pack, it's normal to see a number of tables modified in the SQL Sentry database on a regular basis due to the partitioning. You may want to filter out these tables in the queries. See the SQL Sentry Scalability Pack Tables to Exclude from Index Maintenance.sql query on GitHub to get a complete list of those tables.

Triggers - Created

This condition evaluates to *True* when it finds databases with triggers that have been created with the past hour.

The results returned are number of triggers in a database that meet the condition (**Value**) and the name of the database that has the created triggers (**Key**).

If the condition evaluates to *True*, run the following query against the database(s) in the result set to get a list of all triggers created **today**:

```
SELECT t.[name] as [trigger name],
t.is_instead_of_trigger as [is instead of],
SCHEMA_NAME(s.schema_id) as [parent schema],
OBJECT_NAME(t.parent_id) as [parent object],
o.type_desc as [parent type],
t.create_date
FROM sys.triggers t
JOIN sys.objects o
ON o.object_id = t.parent_id
JOIN sys.schemas s ON o.schema_id = s.schema_id
WHERE t.create_date >= (SELECT CONVERT(DATE,GETDATE()))
ORDER BY t.create_date DESC;
```

An **Execute SQL** action can also be used to automatically email a list to you.

Autogrowth Disabled for Database Files

This condition evaluates to *True* when it finds database files in *sys.master_files* that are not set to auto grow.

The default query checks for rows and log files that are online and not read-only.

```
SELECT [name], growth
FROM sys.master_files
WHERE growth = 0 --fixed file size, no growth
AND [type] < 2 --includes Rows and Log only
AND [state] = 0 --include ONLINE only
AND is_read_only = 0; --exclude files that are read-only
```

Additional Information: See the sys.master_files article on Microsoft Docs for more information on the options in this query to customize to the condition.

s1-team-submitted / backups

There are six backup advisory conditions:

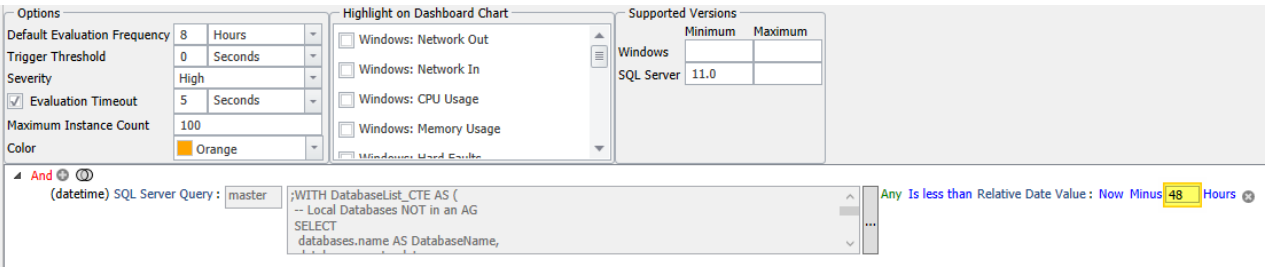
- The ones that have "with AGs" in their name work in environments that have a mix of Availability Groups and no Availability Groups.
    - If you are using the "with AGs" conditions, you do not need to use the other three, unless you have monitored targets that are running versions of SQL Server that are older than 2012.
- The conditions that do not include Availability Groups are appropriate for environments not using Availability Groups, including monitored targets that are on SQL Server versions earlier than 2012.

Database Backup Diff SLA Breached - with AGs

Checks all user databases to find the last *differential* backup for each database.

- Works in environments with and without availability groups
- Restricted to SQL Server 2012 and higher
- Ignores all system databases, distribution, and **ReportServerTempDB** databases
- Ignores *read-only* databases
- Ignores databases in a standby state
- Ignores databases identified as a database snapshot

Default threshold is *48* hours, please change to meet your own service level agreements (SLA).

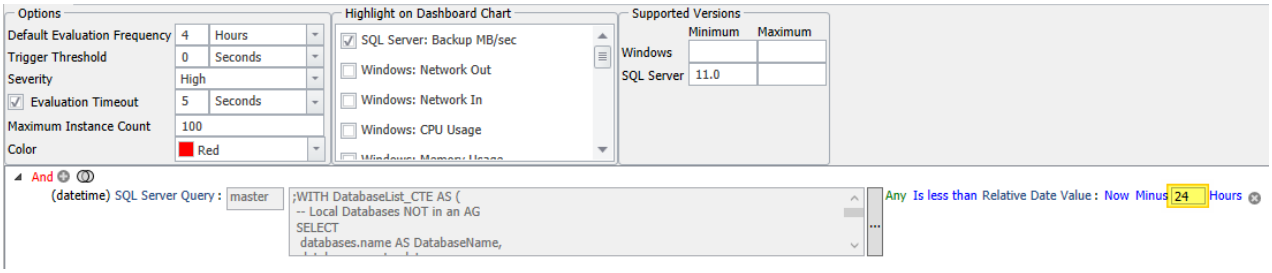


Database Backup Full SLA Breached - with AGs

Checks all user databases to find the last *full* backup for each database.

- Works in environments with and without availability groups
- Restricted to SQL Server 2012 and higher
- Ignores **TempDB** and **ReportServerTempDB** databases
- Ignores *read-only* databases
- Ignores databases in a standby state
- Ignores databases identified as a database snapshot

Default threshold is *24* hours, please change to meet your own service level agreements (SLA).

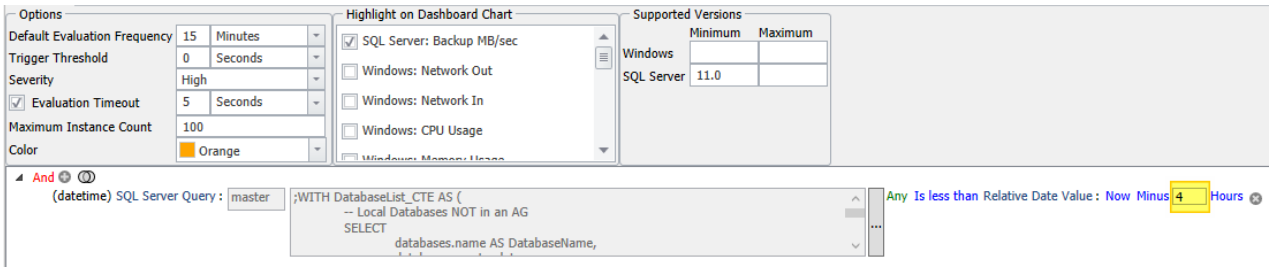


Database Backup Log SLA Breached - with AGs

Checks all user databases to find the last *t-log* backup for each database.

- Works in environments with and without availability groups
- Restricted to SQL Server 2012 and higher
- Ignores **model**, **tempdb**, and **ReportServerTempDB** databases
- Ignores *read-only* databases
- Ignores databases in a standby state
- Ignores databases identified as a database snapshot
- Ignores databases in *simple recovery* mode

Default threshold is *4* hours, please change to meet your own service level agreements (SLA).



Legacy Backup Conditions

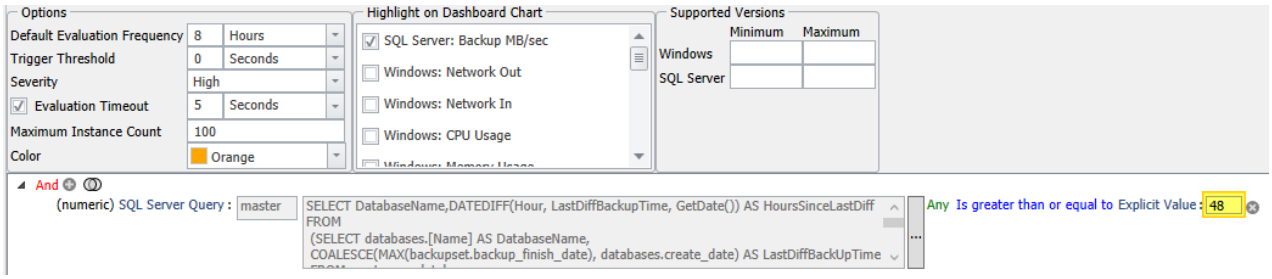
The following versions of the backup conditions that work in older versions of SQL Server:

Database Backup Diff SLA Breached

Checks all user databases to find the last *differential* backup for each database.

- Ignores all system databases, distribution, and **ReportServerTempDB** databases
- Ignores *read-only* databases
- Ignores databases in a standby state
- Ignores databases identified as a database snapshot

Default threshold is *48* hours, please change to meet your own service level agreements (SLA).

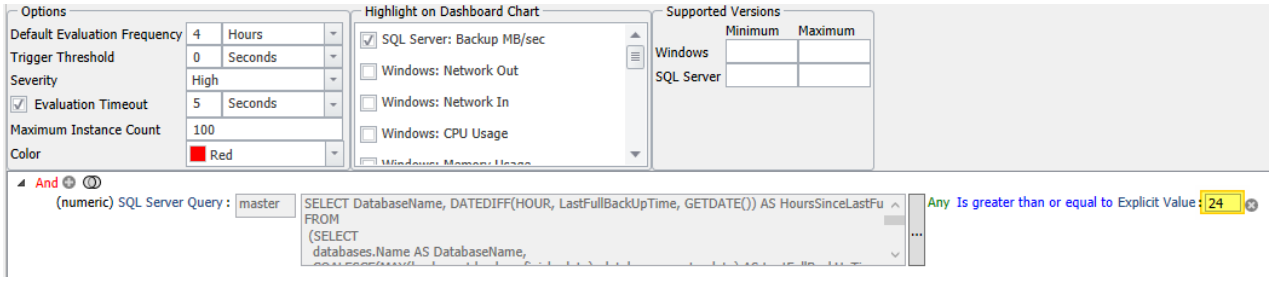


Database Backup Full SLA Breached

Checks all user databases to find the last *full* backup for each database.

- Ignores **TempDB** and **ReportServerTempDB** databases
- Ignores *read-only* databases
- Ignores databases in a standby state
- Ignores databases identified as a database snapshot

Default threshold is *24* hours, please change to meet your own service level agreements (SLA).

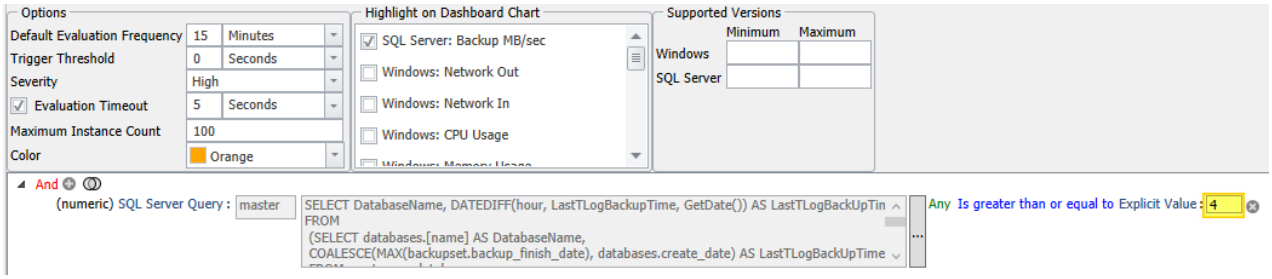


Database Backup Log SLA Breached

Checks all user databases to find the last *t-log* backup for each database.

- Ignores **model**, **tempdb**, and **ReportServerTempDB** databases
- Ignores *read-only* databases
- Ignores databases in a standby state
- Ignores databases identified as a database snapshot
- Ignores databases in *simple recovery* mode

Default threshold is *4* hours, please change to meet your own service level agreements (SLA).



s1-team-submitted / miscellaneous

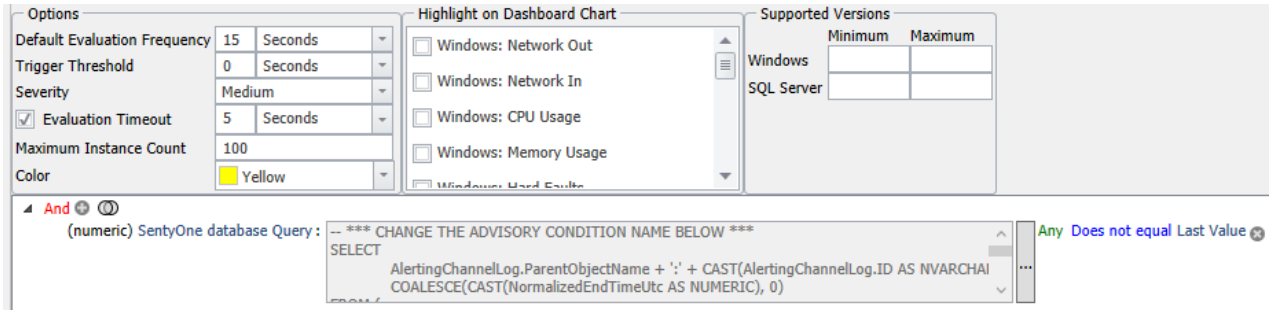
Advisory Condition Cleared (% Free Space - Disk Per Server)

This condition will, per server, monitor whether a given advisory condition that has been met, has changed back to a *cleared* status.

> **Note:** This has been tested with SQL Sentry advisory conditions but not with global advisory conditions, so it may not work as expected against global ones.

> **Important:** You must create a copy per advisory condition you wish to get an *all clear* on. This requires editing the SQL Sentry database Query as well, and substituting the name of the advisory condition in the condition name and query. In this example, the name used is **% Free Space - Disk Per Server**.

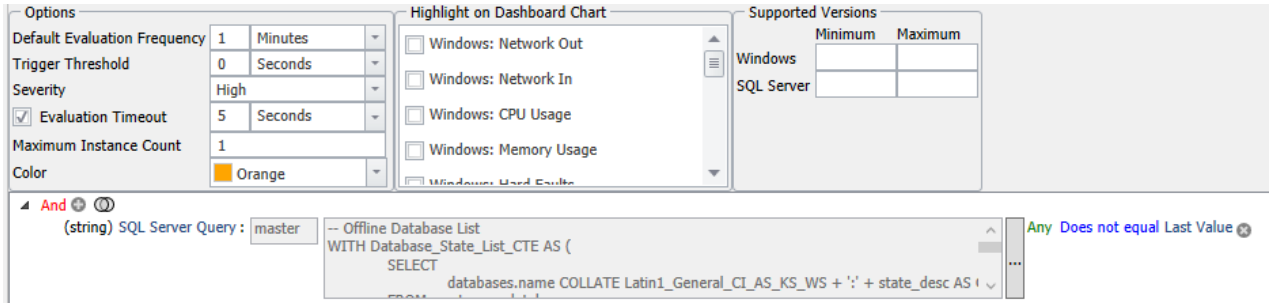
```
-- *** CHANGE THE ADVISORY CONDITION NAME BELOW ***
SELECT
 AlertingChannelLog.ParentObjectName + ':' + CAST(AlertingChannelLog.ID AS NVARCHAR(800)),
 COALESCE(CAST(NormalizedEndTimeUtc AS NUMERIC), 0)
FROM (
 -- Make sure we always pull the most recent AlertingChannelLog log entry for a given
 -- server/advisory condition combination
 SELECT
  MAX(AlertingChannelLog.ID) AS MaxAlertingChannelLogID,
  AlertingChannelLog.ObjectID
 FROM dbo.DynamicConditionDefinition
 INNER JOIN dbo.AlertingChannelLog
  ON AlertingChannelLog.DynamicConditionID = DynamicConditionDefinition.ID
 -- *** CHANGE THE ADVISORY CONDITION NAME BELOW ***
 WHERE DynamicConditionDefinition.Name = '% Free Space - Disk Per Server'
 GROUP BY AlertingChannelLog.ObjectID
) AS MaxAlertingChannelLog
INNER JOIN dbo.AlertingChannelLog
 ON AlertingChannelLog.ID = MaxAlertingChannelLog.MaxAlertingChannelLogID
INNER JOIN dbo.EventSourceConnection
 ON AlertingChannelLog.ObjectID = EventSourceConnection.ObjectID
WHERE EventSourceConnection.ID = @ConnectionID
```



Database State - Offline

Lists all databases that are in an offline state.

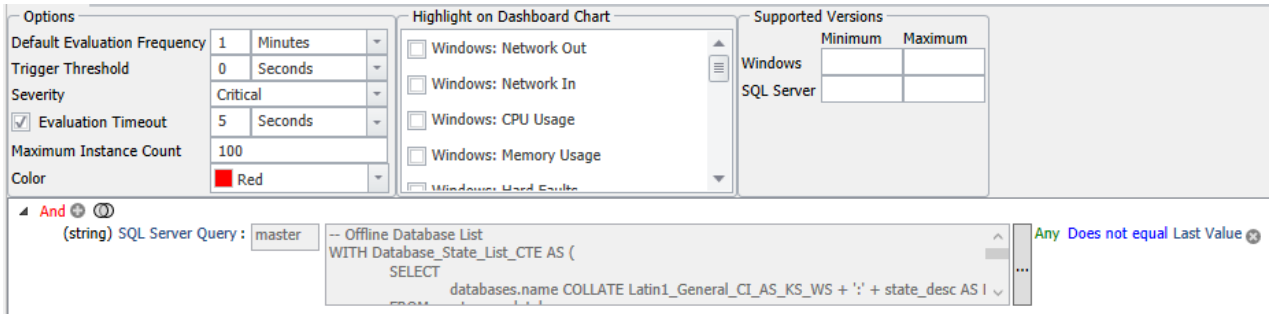
- Condition is met if the list changes.
- On some servers, having databases offline is expected.
    - Review the CSV output list of databases offline to see if there is a new result.
- Also checks databases that are part of an availability group.



Database State - Suspect or Emergency

Lists all databases that are in a Suspect and/or Emergency state.

- Condition is met if the list changes.
    - Review the CSV output list of databases.
- Also checks databases that are part of an availability group.

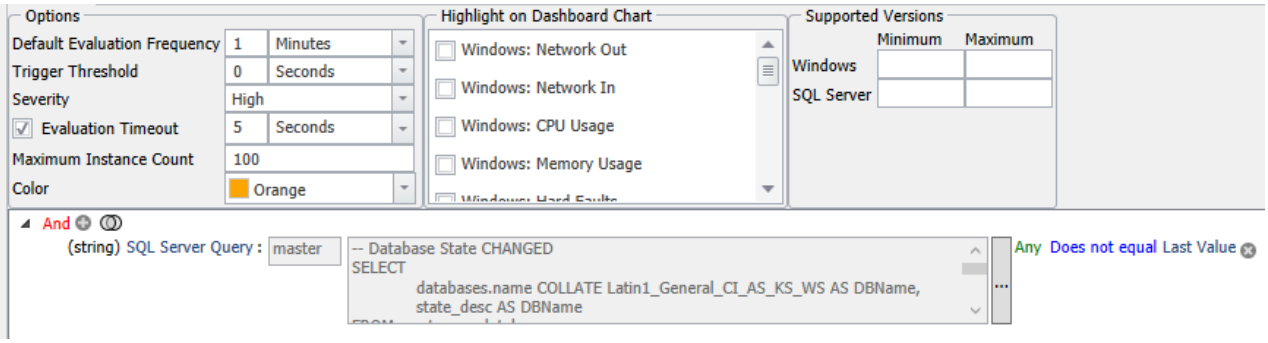


Database State - Changed

Detects if a database has changed state from the state it was in at the prior evaluation.

Note: This applies for any state change, so will alert for databases that are come **ONLINE**, are intentionally taken **OFFLINE**, and so on. You can adjust this behavior to filter out **database states** that are not relevant to you, by adding a predicate filter to the SQL Server Query (see the other **Database State...** conditions for examples).

Additional Information: See the sys.databases article on Microsoft Docs for a complete list of the **state** values.

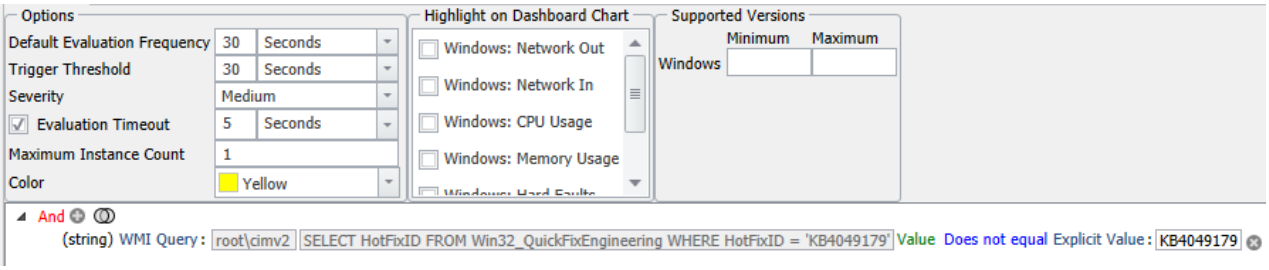


Has KB Been Installed

Example advisory condition to check if a specific KB or hot-fix was applied to all SQL Servers.

Substitute the KB of interest into WMI Query and **Explicit Value**. This example uses *KB4049179*.

```
SELECT HotFixID FROM Win32_QuickFixEngineering WHERE HotFixID = 'KB4049179'
```

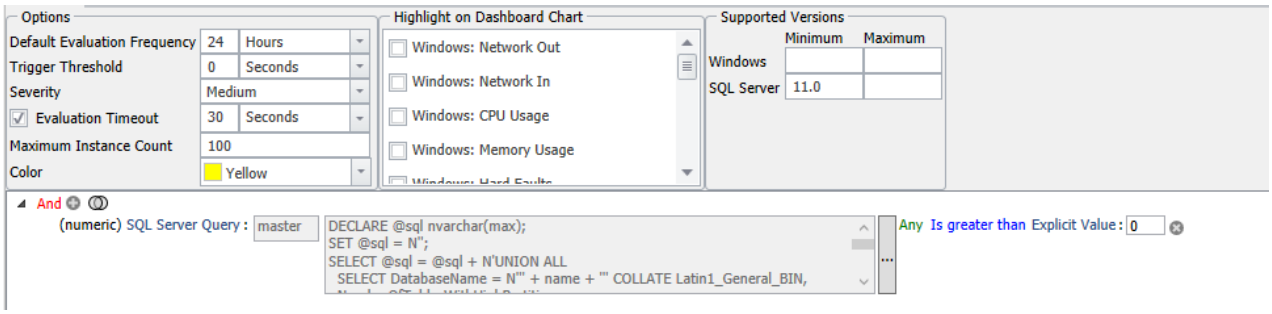


Tables Approaching Maximum Number of Table Partitions

This condition evaluates to *True* when it finds databases with tables that have a number of partitions that is greater than *14,500*. Current versions of SQL Server support 15,000 partitions maximum, and versions prior to 2012 support up to 1,000. Current versions of SQL Server may experience performance issues when there are more than 1,000 partitions (as explained in the Microsoft Docs article on Partitioned Tables and Indexes).

The results returned are number of tables in a database that meet the condition (Value) and the name of the database that has the tables (Key). If the condition evaluates to True, run the following query against the database(s) in the result set to get a list of all tables meeting the condition and the number of partitions in each:

```
SELECT DISTINCT [t].[name], [p].[partition_number]
FROM sys.partitions p WITH (NOLOCK)
INNER JOIN sys.tables t WITH (NOLOCK)
ON p.object_id = t.object_id
WHERE [p].[partition_number] = (SELECT MAX([sp].[partition_number])
FROM sys.partitions sp WITH (NOLOCK)
WHERE [sp].[object_id] = [t].[object_id]
AND [sp].[partition_number] > 14500)
AND [p].[partition_number] > 14500;
```



Note:

- This returns up to 100 databases in the alert (this is the **Maximum Instance Count**).
- It uses SQL Server 2012 as the **Minimum** version (*11.0*) which allows for up to 15,000 partitions. If you are monitoring SQL Servers on lower versions, you may want to copy/change this condition to use lower versions and change the *14500* value in the query to something less than 1,000.
- Adjust the *14500* value in the **SQL Server Query** depending on the needs of your environment and how often new partitions are created.
- This is set to run daily (every 24 hours) in the **Default Evaluation Frequency** and that may need to be adjusted based on how often new partitions are created, as well as the **Severity** level.
- If your primary concern is potential performance issues associated with having more than 1,000 partitions, consider creating a clone of this condition and updating the name/settings to reflect that scenario.

s1-team-submitted / scalability-pack

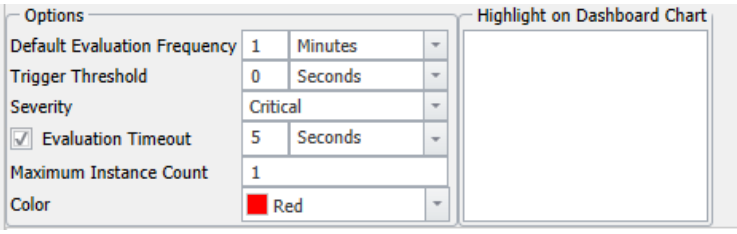
These conditions are only for SQL Sentry databases that have the SQL Sentry Scalability Pack installed. While the default scalability pack settings work for most environments, it's important that you contact the support team if any errors surface related to the scalability pack to ensure that your settings are configured for optimal performance and data flow.

CCI Partitioning Errors Exist

Contact our support team for assistance. Query the [Partitioning].[ActionErrorLogging] table to view error details. Table holds 7 days worth of information.

To view error messages:

```
SELECT * FROM [Partitioning].[ActionErrorLogging]
WHERE [Action] = 0
ORDER BY ActionTime DESC;
```



Note: Errors are often caused by enabling index maintenance jobs or statistics updates against tables that should be excluded.

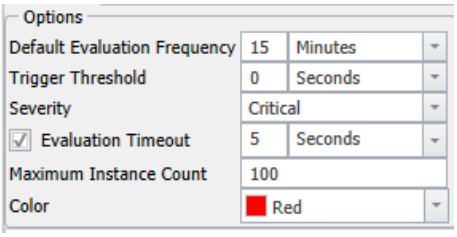
This usually surfaces as an error message like *Transaction (Process ID 123) was deadlocked on lock resources with a...* related to a process such as *ALTER PARTITION SCHEME [PerfromanceDataCurrentScheme] NEXT USED [Primary]* and statements such as *SELECT StatMan([SC0]) FROM (SELECT TOP 100 PERCENT [DeviceID] AS [SC0] FROM [dbo].[PerfromanceDataAnalysis]* showing up in Top SQL around the same time as the error.

Any tables found by the **SQL Sentry Scalability Pack Tables to Exclude from Index Maintenance.sql** script on GitHub should be excluded from any index maintenance you perform against the SQL Sentry database when the scalability pack is installed.

High Number of Table Partitions in SentryOne Database

Evaluates to *True* when there are 10,000 or more partitions in a table in the SQL Sentry database. Microsoft SQL Server has a limit of 15,000 partitions per table.

Contact our support team to determine if there are configuration changes needed to adjust the rate of table partitioning for this environment.



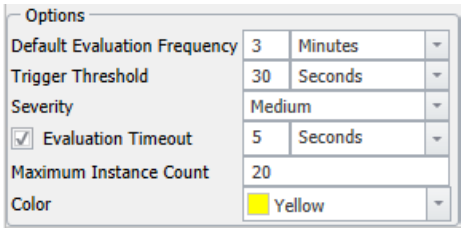
High Rows in In-memory Staging Table

Returns all in-memory performance data staging tables with an unusually high number of rows. Default=500,000, but this may need to be adjusted per environment.

When the number of rows is over the threshold, it means that the **MovePABufferData** procedure is not keeping up with the volume of incoming performance data, and thus is not moving data from the in-memory tables into the CCI tables quickly enough.

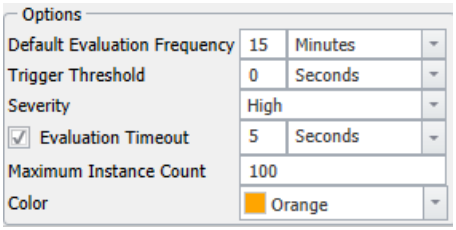
This can be caused by:

1. High latency network links between the SQL Sentry monitoring service(s) and the SQL Sentry database which is causing the performance data for some targets to be written too slowly. The **MovePABufferData** process will wait up to 60 seconds by default for committing each new timestamp, and continual lag may lead to an accumulation of staging data. Contact our support team for assistance troubleshooting slow links.
2. The **MovePABufferData** process is not running -- the associated thread may have experienced an error and not restarted successfully. Restart the primary SQL Sentry monitoring service to reset it -- this is the one with with the oldest **Last Initialized** time under **All Targets** -> **Show Monitoring Services List**.
3. Transient maintenance activities like index rebuilds, backups, etc., which can cause resource contention and/or blocking leading to temporary queuing of the staging data. The max value can be adjusted to account for these spikes, or a Response Ruleset can be used to wait for *n* minutes before alerting.



Partition with High Number of Rows

Evaluates to *True* on columnstore partitions with more than 20 million rows in the SQL Sentry database. Please contact our support team if this happens as the partitioning configuration options will need to be analyzed and perhaps adjusted for optimal performance.

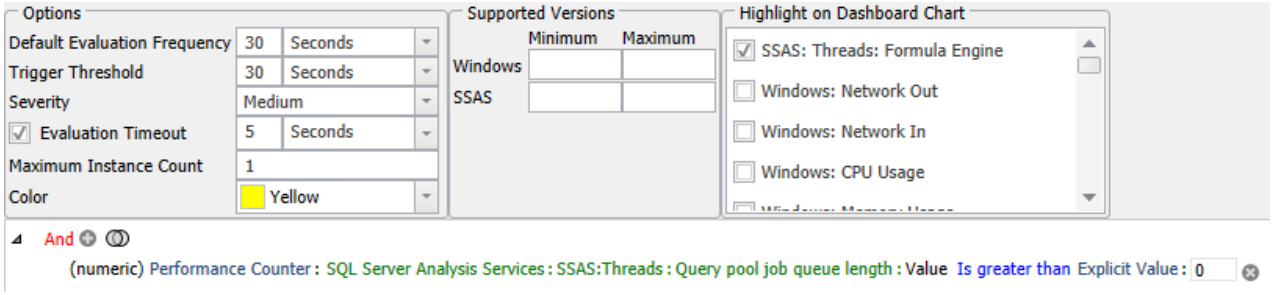


s1-team-submitted / ssas

SSAS Formula Engine Query Pool Job Queuing

The query pool refers to **Formula Engine** activity for queries. If you're seeing consistently high queue lengths, but not high CPU utilization, adjust your **MaxThreads** and/or **CoordinatorExecutionMode** properties for your SSAS instance.

Remember that the **Formula Engine** is single threaded, so increasing the query pool setting may not improve performance of any one query, but it may improve the performance in handling multiple simultaneous requests.

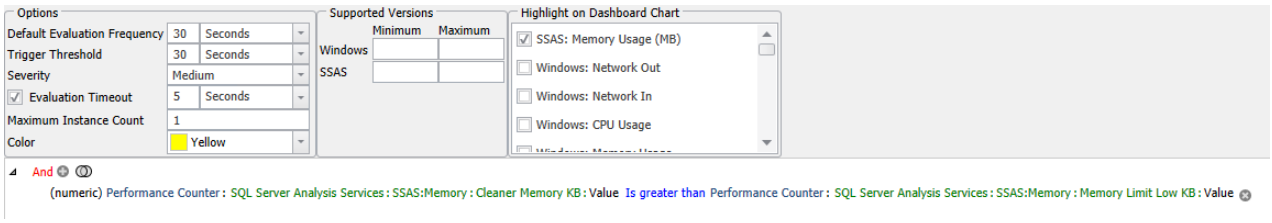


Additional Information: See section 6.11 of the SSAS 2008 Performance Guide.

SSAS High Memory Limit Exceeded

SSAS uses memory limit settings to determine how it allocates and manages its internal memory. Memory\LowMemoryLimit defaults to 65 percent of the total available physical memory on the machine (75 percent on AS2005), and Memory\TotalMemoryLimit (also sometimes called the High Memory Limit) defaults to 80 percent. This is the total amount of memory that the SSAS process itself (msmdsrv.exe) can consume.

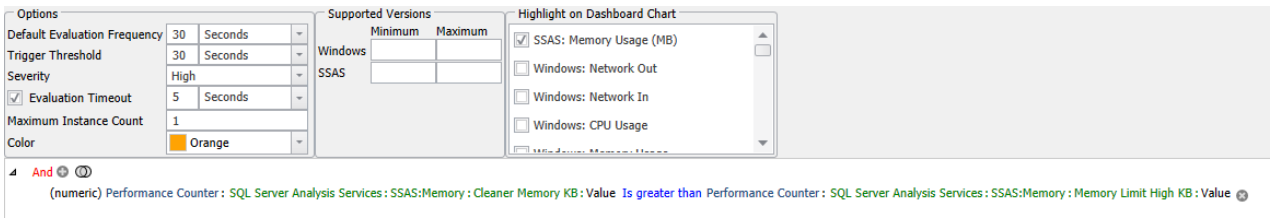
Once memory usage hits the low limit, the memory cleaner threads starts moving data out of memory. If memory hits the total limit, the cleaner goes into crisis mode. It spawns additional threads and gets more aggressive about memory cleanup, and this can dramatically impact performance.



SSAS Low Memory Limit Exceeded

SSAS uses memory limit settings to determine how it allocates and manages its internal memory. Memory\LowMemoryLimit defaults to 65% of the total available physical memory on the machine (75% on AS2005), and Memory\TotalMemoryLimit (also sometimes called the High Memory Limit) defaults to 80 percent. This is the total amount of memory that the SSAS process itself (*msmdsrv.exe*) can consume.

Once memory usage hits the low limit, the memory cleaner threads start moving data out of memory. If memory hits the total limit, the cleaner goes into crisis mode. It spawns additional threads and gets more aggressive about memory cleanup, and this can dramatically impact performance.

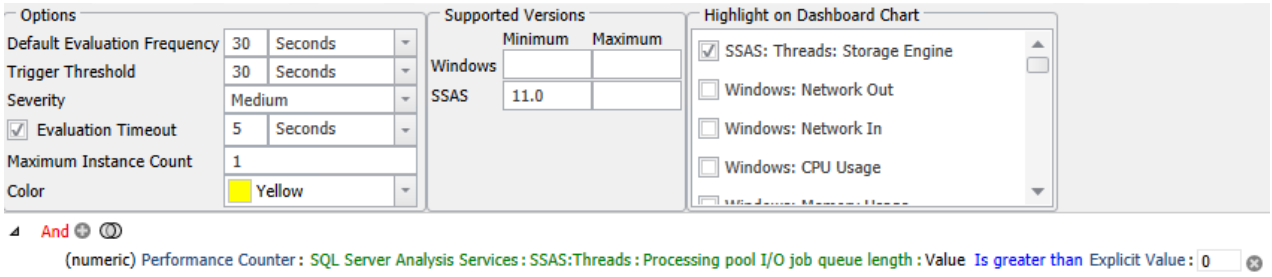


SSAS Storage Engine IO Job Queuing

The IOProcess thread pool separates reads from other activities. If the I/O job queue length is consistently

above zero, you may be experiencing an I/O bottleneck.

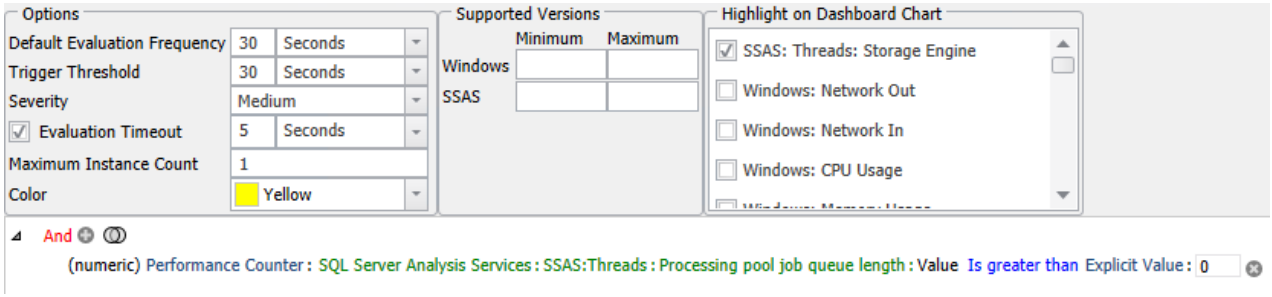
For more information, see the Analysis Services MOLAP Guide for SQL Server 2012 and 2014.



SSAS Storage Engine Processing Pool Job Queuing

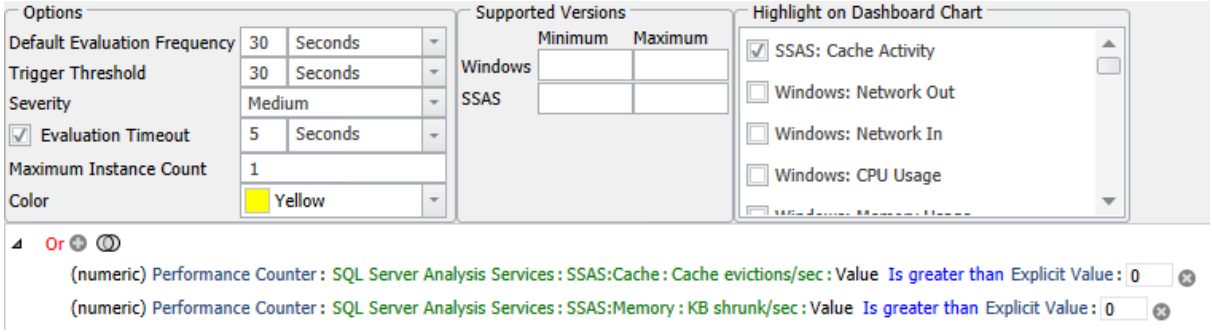
Depending on your version of SSAS, queuing of jobs in this pool can be related to all **Storage Engine** activity (SSAS 2005 to 2008R2), or strictly processing activity in SSAS 2012 and above.

For more information on optimizing this activity for your version of SSAS, see the appropriate **Microsoft SSAS Performance Guide**.



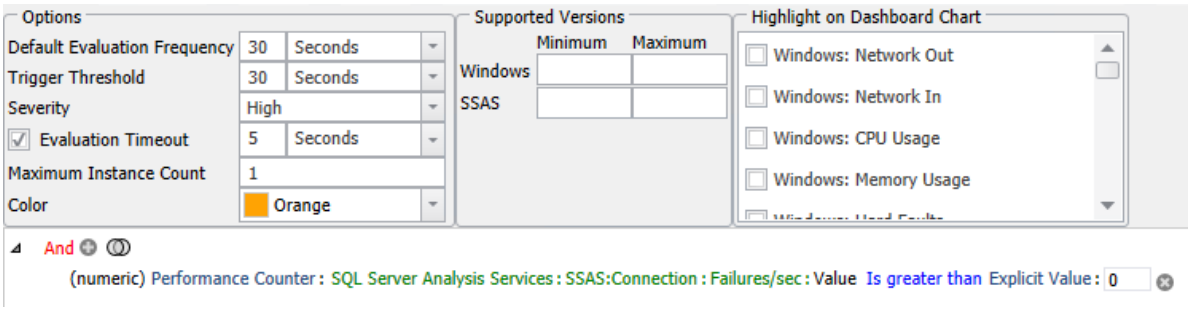
SSAS Sustained Cache Evictions

If **Cache Evictions/sec** or **Memory : KB shrunk/sec** are consistently above zero, you may have memory pressure on the SSAS instance. This is often seen when SSAS memory usage exceeds configured limits.



SSAS Sustained Connection Failures

A sustained value above zero indicates an inability for users to successfully connect to SSAS. This could be related to overburdened resources on the server.

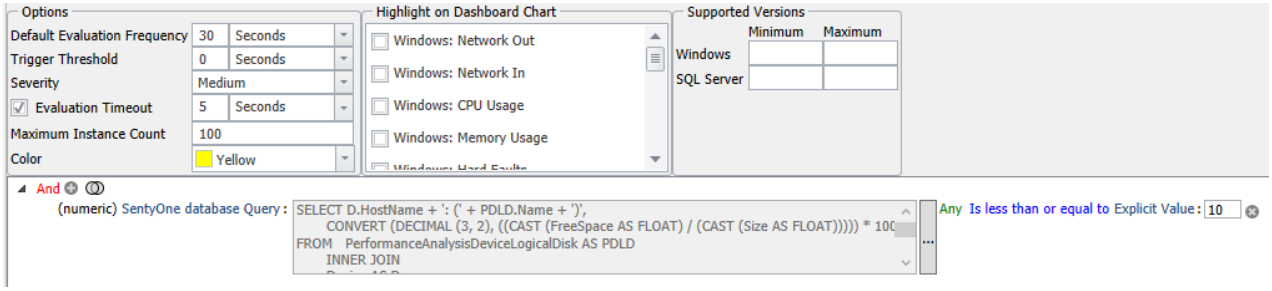


s1-team-submitted / storage

% Free Space - Disk Per Server

A SQL Sentry database Query that returns all disks (per monitored server) with less than 10% free space. The alert format is returned as follows: **Key (server name)**, **value (% free space)**.

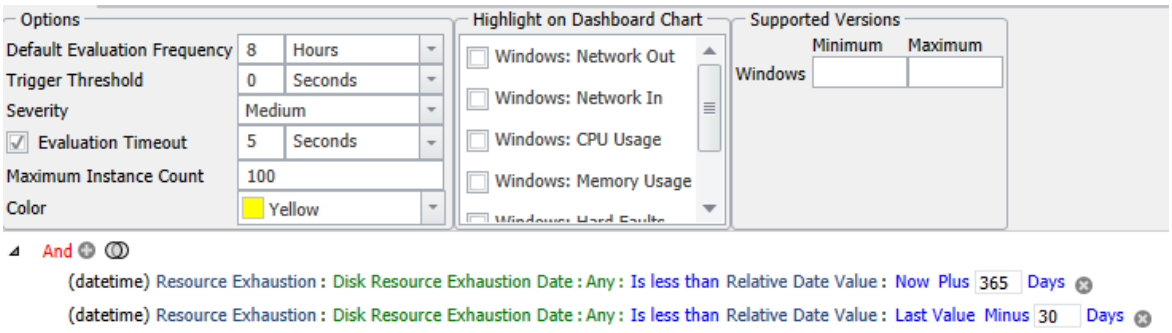
> **Note:** The type is SQL Server, not SQL Sentry (or SentryOne), which is useful if you wish to set different thresholds on different types of servers (*dev* vs. *prod*).



Windows Volume Exhaustion Date Changed

This condition evaluates to **True** if any forecasted exhaustion dates for any drives are within one year of the current date and have had that date change to be at least 30 days earlier than previously forecast.

This can help identify situations where a new database has been deployed, or a workload changed, possibly leading to accelerated database growth. That might lead to a much earlier exhaustion date, without you being caught by surprise.



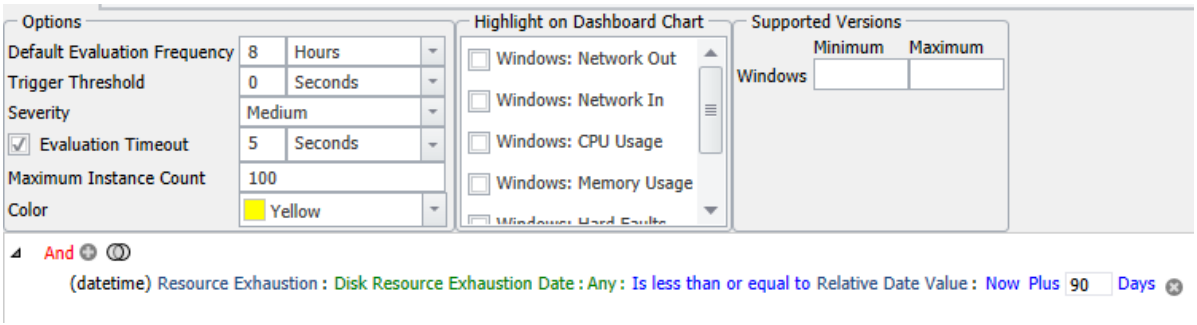
Additional Information: Introducing Storage Forecasting with SQL Server Machine Learning Services

Windows Volume Forecasted Exhaustion Within 90 Days

This condition checks the **Resource Exhaustion Dates** (REDs) for all forecasted volumes and evaluates to **True** if any occur within the next 90 days.

The value can be extended past 90 days, but note the REDs will only be provided if they are calculated within the forecast range, which is 180 days by default.

This condition returns up to 100 records, which is the maximum configurable for any advisory condition.



Additional Information: Introducing Storage Forecasting with SQL Server Machine Learning Services