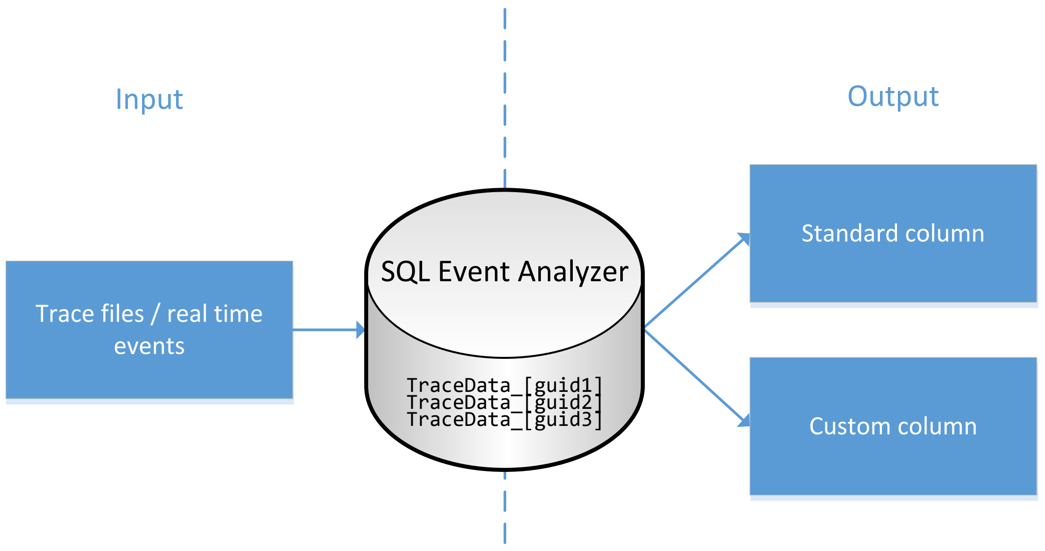
SQL Event Analyzer

Content

1. Introduction

SQL Event Analyzer makes it possible to show and work with events on a SQL Server.

Events can be retrieved from trace files or real time events.



When trace files or real time events are imported, they will be imported to a table on the SQL Server. This table is from now on referred to as a "TraceData table". A TraceData table has the following naming format:

```
TraceData_[Session Id]
```

where `Session Id` is a guid.

Example: `TraceData_f63631c1-5824-4dce-b631-f45e187c3cc1`

Data stored in a TraceData table will be referred to as a "session". This means, that a session can contain one or more imported trace files or real time events.

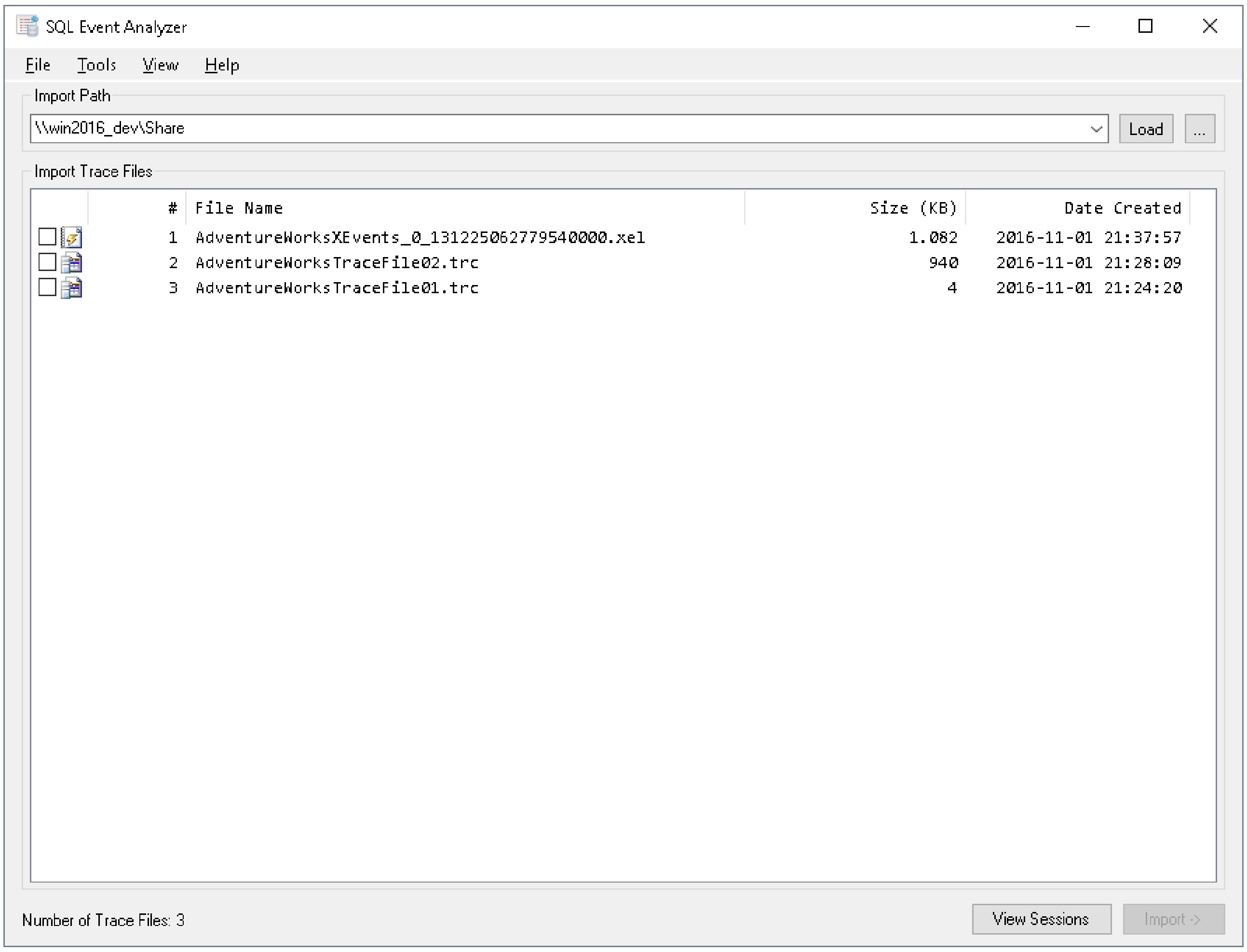
With a TraceData table as input, the events will be presented as output in a list view containing columns. Output columns consists of one or more standard columns optionally combined with custom columns.

Trace files in the following formats are supported:

- SQL Server Profiler
- Extended Events

Note, Extended Events are not supported on SQL Server versions older than SQL Server 2012.

Example of input from trace files:

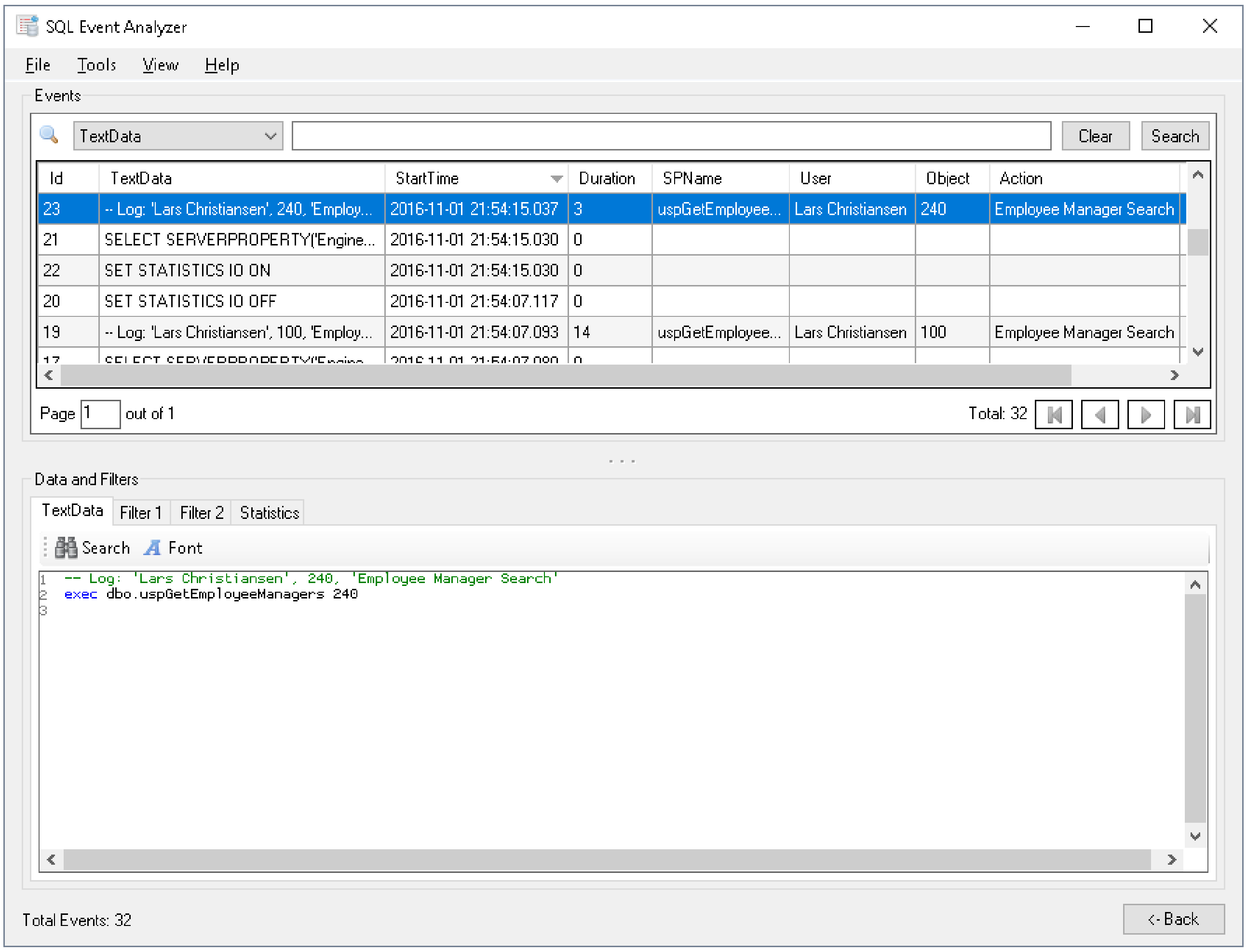


\\win2016_dev\Share

	#	File Name	Size (KB)	Date Created
☐	1	AdventureWorksXEvents_0_131225062779540000.xel	1.082	2016-11-01 21:37:57
☐	2	AdventureWorksTraceFile02.trc	940	2016-11-01 21:28:09
☐	3	AdventureWorksTraceFile01.trc	4	2016-11-01 21:24:20

Number of Trace Files: 3

Example of how the output is shown:


SQL Event Analyzer

File Tools View Help

Events

TextData | Clear | Search

Id	TextData	StartTime	Duration	SPName	User	Object	Action
23	-- Log: 'Lars Christiansen', 240, 'Employ...	2016-11-01 21:54:15.037	3	uspGetEmployee...	Lars Christiansen	240	Employee Manager Search
21	SELECT SERVERPROPERTY('Engine...	2016-11-01 21:54:15.030	0				
22	SET STATISTICS IO ON	2016-11-01 21:54:15.030	0				
20	SET STATISTICS IO OFF	2016-11-01 21:54:07.117	0				
19	-- Log: 'Lars Christiansen', 100, 'Employ...	2016-11-01 21:54:07.093	14	uspGetEmployee...	Lars Christiansen	100	Employee Manager Search

Page 1 out of 1 Total: 32

Data and Filters

TextData Filter 1 Filter 2 Statistics

Search Font

```
-- Log: 'Lars Christiansen', 240, 'Employee Manager Search'
exec dbo.uspGetEmployeeManagers 240
```

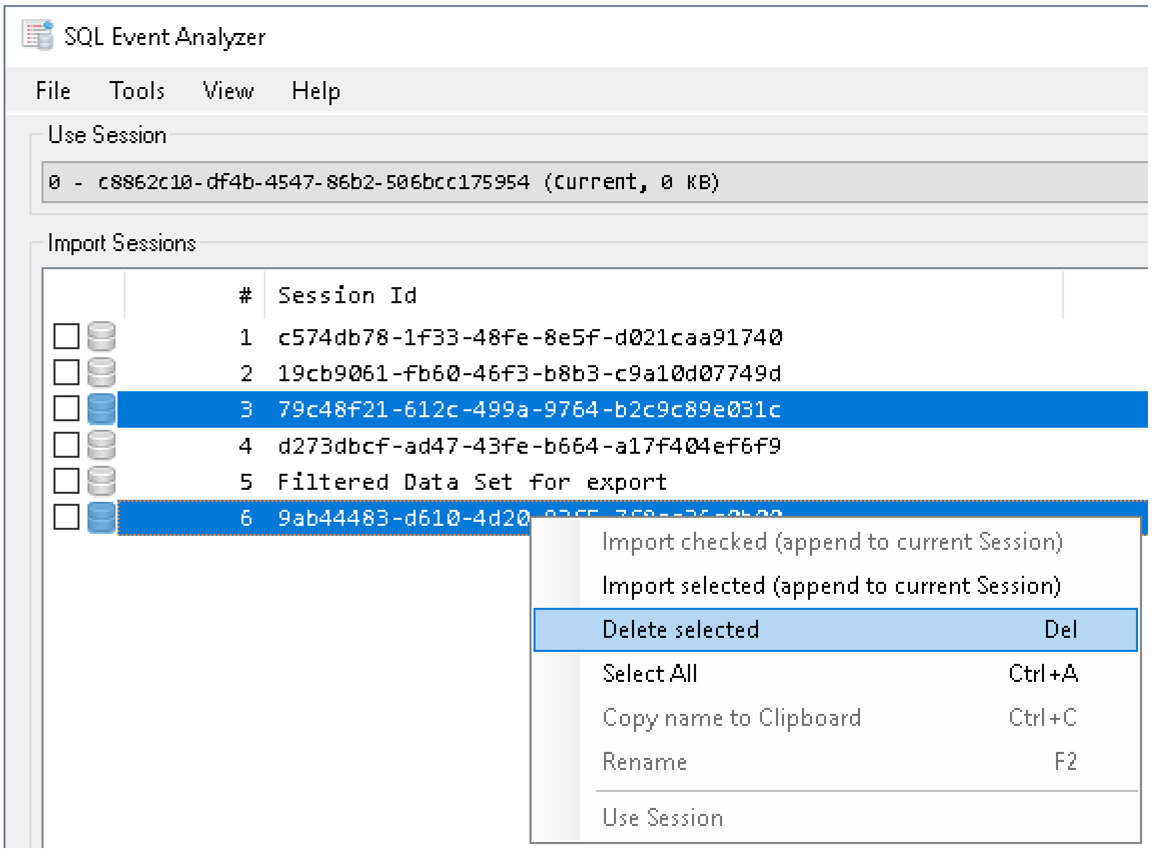
Total Events: 32 <- Back

When running SQL Event Analyzer for the first time, a database on the SQL Server with the name "SQLEventAnalyzer" will be created. If the database already exists, this database will be used.

The SQLEventAnalyzer database contains the TraceData tables.

When SQL Event Analyzer closes, it can be chosen to automatically delete the TraceData tables. By default, the TraceData tables will not be deleted, so it is possible to use a already existing sessions with imported data.

Saved sessions can manually be deleted from the session list:

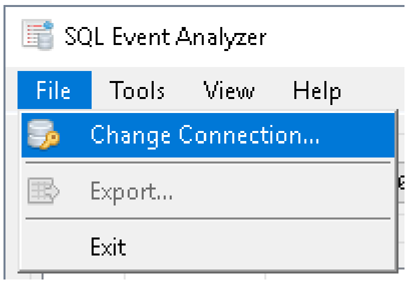


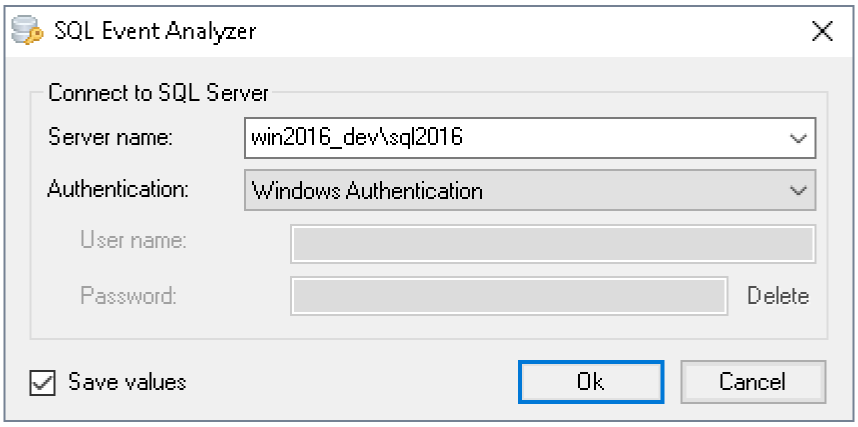
```
0 - c8862c10-df4b-4547-86b2-506bcc175954 (Current, 0 KB)
```

```
   #  Session Id
   1  c574db78-1f33-48fe-8e5f-d021caa91740
   2  19cb9061-fb60-46f3-b8b3-c9a10d07749d
   3  79c48f21-612c-499a-9764-b2c9c89e031c
   4  d273dbcf-ad47-43fe-b664-a17f404ef6f9
   5  Filtered Data Set for export
   6  9ab44483-d610-4d20
```

Import checked (append to current Session)
Import selected (append to current Session)
Delete selected Del
Select All Ctrl+A
Copy name to Clipboard Ctrl+C
Rename F2
Use Session

If SQL Event Analyzer is set to delete sessions upon exit, this can be ignore by holding down the Shift key when SQL Event Analyzer closes. The same applies when changing to a new connection from within the application, then holding down the Shift key while choosing "Ok" will keep the session:




SQL Event Analyzer

Connect to SQL Server

Server name: win2016_dev\sql2016

Authentication: Windows Authentication

User name:

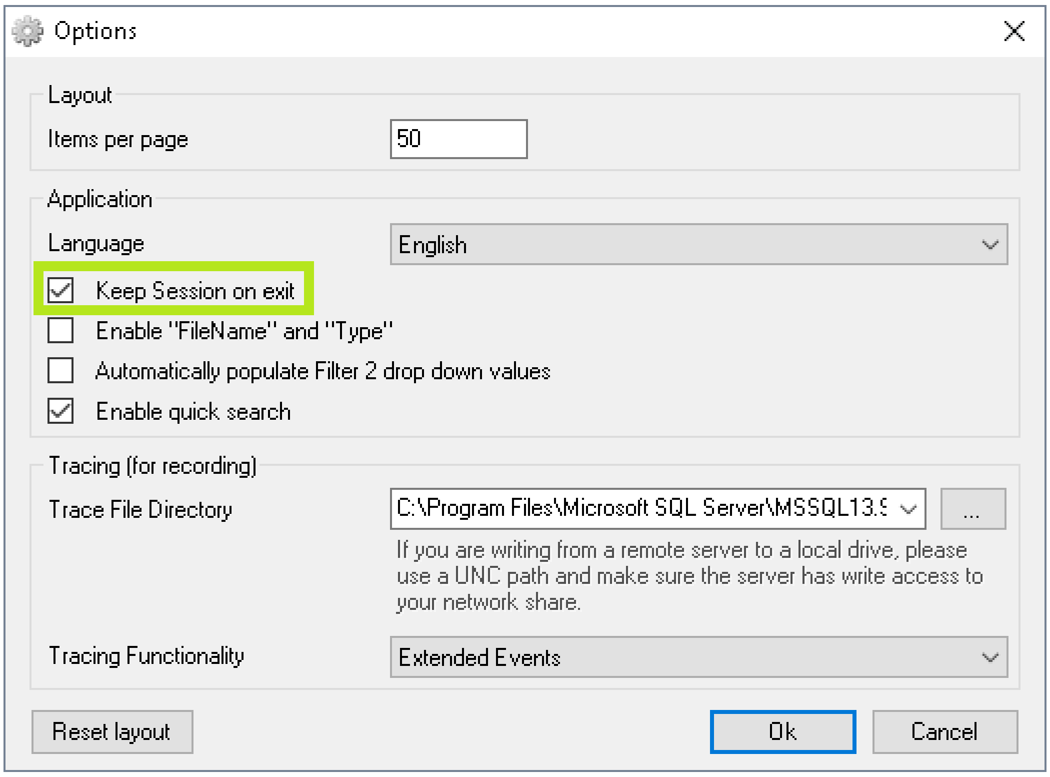
Password:

Delete

☑ Save values

Ok Cancel

If sessions should be kept or deleted when closing SQL Event Analyzer can be set in the "Options" menu:


Options

Layout

Items per page: 50

Application

Language: English

☑ Keep Session on exit

☐ Enable "FileName" and "Type"

☐ Automatically populate Filter 2 drop down values

☑ Enable quick search

Tracing (for recording)

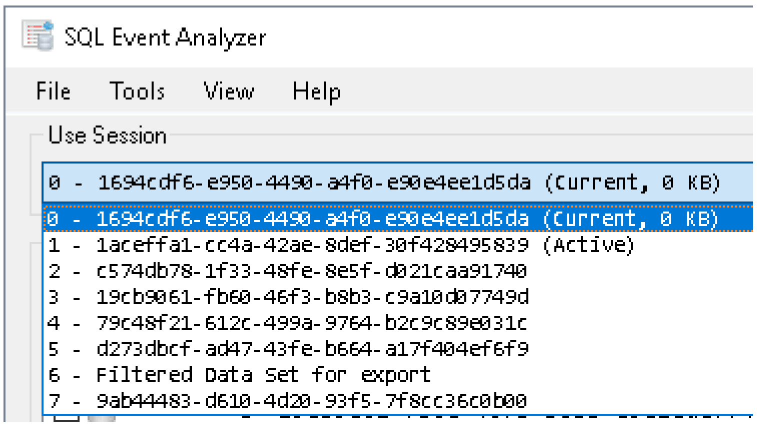
Trace File Directory: C:\Program Files\Microsoft SQL Server\MSSQL13.S ...

If you are writing from a remote server to a local drive, please use a UNC path and make sure the server has write access to your network share.

Tracing Functionality: Extended Events

Reset layout   Ok   Cancel

If another user is using a session, the session will be marked as "Active" in the session drop down list:



```
0 - 1694cdf6-e950-4490-a4f0-e90e4ee1d5da (Current, 0 KB)
0 - 1694cdf6-e950-4490-a4f0-e90e4ee1d5da (Current, 0 KB)
1 - 1aceffa1-cc4a-42ae-8def-30f428495839 (Active)
2 - c574db78-1f33-48fe-8e5f-d021caa91740
3 - 19cb9061-fb60-46f3-b8b3-c9a10d07749d
4 - 79c48f21-612c-499a-9764-b2c9c89e031c
5 - d273dbcf-ad47-43fe-b664-a17f404ef6f9
6 - Filtered Data Set for export
7 - 9ab44483-d610-4d20-93f5-7f8cc36c0b00
```

2. Permissions

The user, that connects to the SQL Server where SQL Event Analyzer operates, must have the following permissions:

- "db_owner" rights to the "SQLEventAnalyzer" database
- "Alter any connection" rights on server level

If using SQL Server 2008 R2 or older, the user must have the following permission:

- "Alter trace" rights on server level

If using SQL Server 2012 or newer, the user must have the following permissions:

- "Alter trace" rights on server level
- "View server state" rights on server level

Refer to the "Changing tracing functionality" section for more information about tracing functionality.

Note, if the user account that uses SQL Event Analyzer is not member of the local administrators group, the following will not be saved:

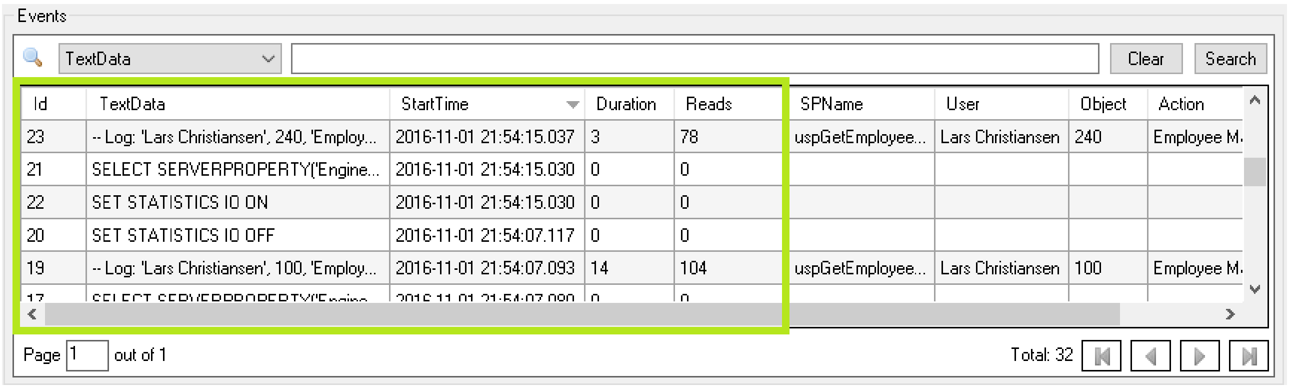
- Settings
- Saved searches
- Changes in the user interface

3. Standard columns

Events will by default always include the following data:

Id:	Automatically generated increasing Id number
TextData:	The executed SQL statement
FileName:	The trace file containing the event
Type:	The event type
SPID:	The SPID for the event
Duration:	The execution time for the event in milliseconds
StartTime:	Start time for the event
Reads:	Number of logical reads for the event
Writes:	Number of writes for the event
CPU:	The total CPU time for the event in milliseconds
Rows:	Number of rows worked on in the event

The above data will be shown as standard columns in the output list view:



Id	TextData	StartTime	Duration	Reads	SPName	User	Object	Action
23	-- Log: 'Lars Christiansen', 240, 'Employ...	2016-11-01 21:54:15.037	3	78	uspGetEmployee...	Lars Christiansen	240	Employee M...
21	SELECT SERVERPROPERTY('Engine...	2016-11-01 21:54:15.030	0	0				
22	SET STATISTICS IO ON	2016-11-01 21:54:15.030	0	0				
20	SET STATISTICS IO OFF	2016-11-01 21:54:07.117	0	0				
19	-- Log: 'Lars Christiansen', 100, 'Employ...	2016-11-01 21:54:07.093	14	104	uspGetEmployee...	Lars Christiansen	100	Employee M...
17	SELECT SERVERPROPERTY('Engine...	2016-11-01 21:54:07.090	0	0				

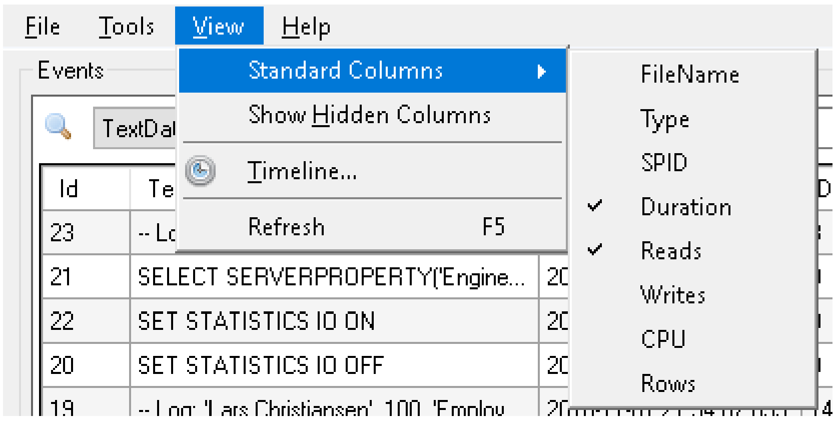
Page 1 out of 1

Total: 32

Note, the following columns are not shown by default:

- FileName
- Type
- SPID
- Writes
- CPU
- Rows

Which standard columns to be shown can be chosen in "View", "Standard Columns":



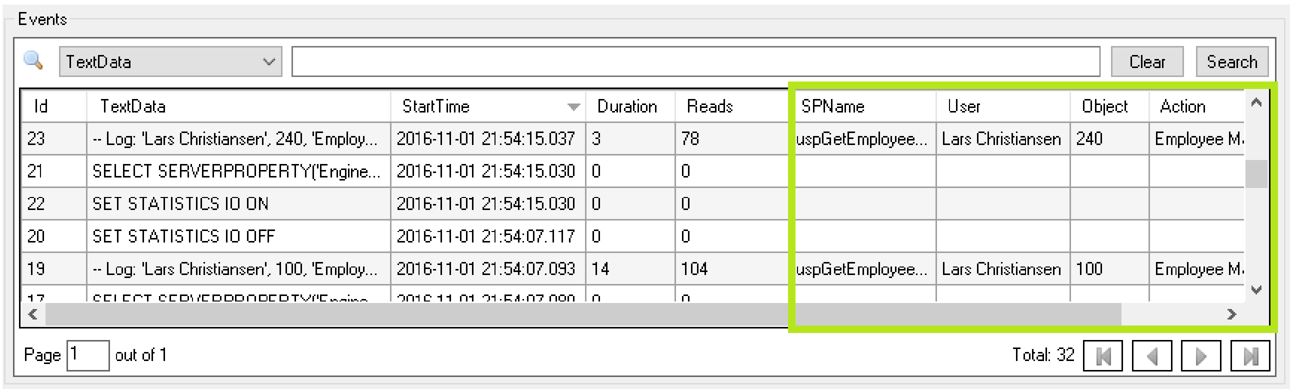
It is not possible to hide the "Id", "TextData" and "StartTime" columns.

A standard column can be used elsewhere in SQL Event Analyzer by using its name. I.e. it is possible to use a standard column in a custom column.

4. Custom columns

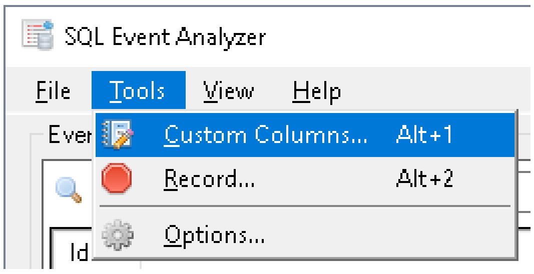
It is possible to add extra custom columns besides from the standard columns.

Custom columns are by default shown to the right of the standard columns:



Id	TextData	StartTime	Duration	Reads	SPName	User	Object	Action
23	-- Log: 'Lars Christiansen', 240, 'Employ...	2016-11-01 21:54:15.037	3	78	uspGetEmployee...	Lars Christiansen	240	Employee M.
21	SELECT SERVERPROPERTY('Engine...	2016-11-01 21:54:15.030	0	0				
22	SET STATISTICS IO ON	2016-11-01 21:54:15.030	0	0				
20	SET STATISTICS IO OFF	2016-11-01 21:54:07.117	0	0				
19	-- Log: 'Lars Christiansen', 100, 'Employ...	2016-11-01 21:54:07.093	14	104	uspGetEmployee...	Lars Christiansen	100	Employee M.

Custom columns can be managed from "Tools", "Custom Columns...":

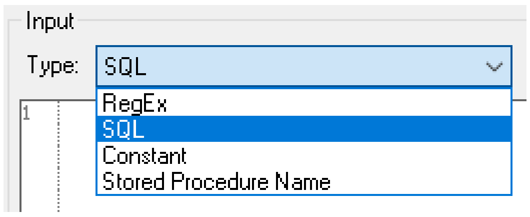


In a custom column, an input criteria can be set. If the input criteria is met, the corresponding output will be shown in the custom column.

A custom column must have an input type and an output type set.

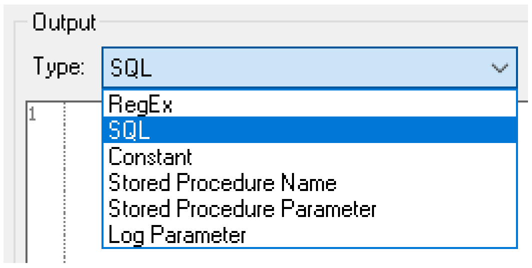
An input type can be:

- RegEx
- SQL
- Constant
- Stored Procedure name



An output type can be:

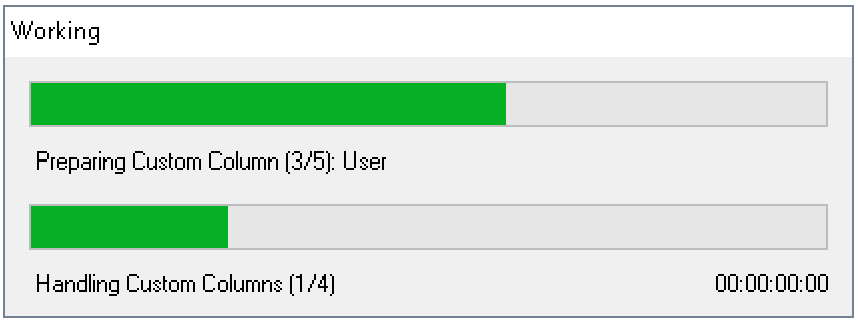
- RegEx
- SQL
- Constant
- Stored Procedure name
- Stored Procedure parameter
- Log parameter



The name of the custom column must be unique, and can be used elsewhere in SQL Event Analyzer. I.e. it is possible to use a custom column from another custom column. The name for the custom column can not be the same name as the name for a statistics column, e.g. "AvgDuration" (see the "Statistics" section for more information). Likewise, the name for a custom column can not be the same as the name of a standard column, e.g. "Reads".

4.1 Generating output from custom columns

Before events can be shown in the output list view, the events must be handled by the values given in the custom columns. If the custom column values are changed, the output list view will be generated again.

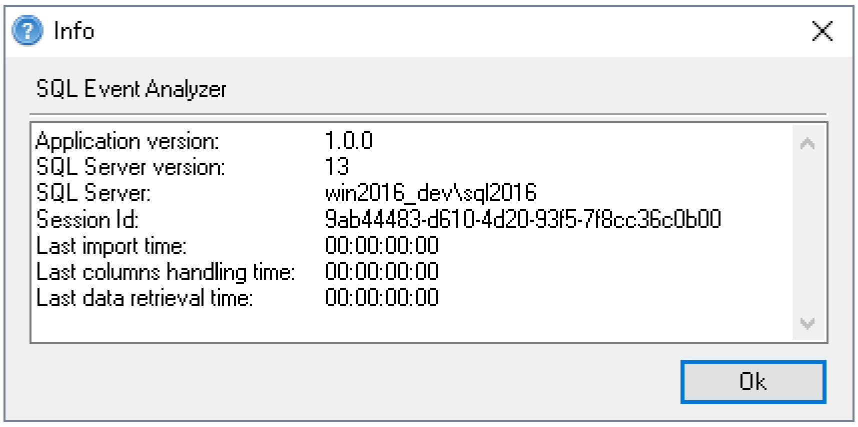


Preparing Custom Column (3/5): User

Handling Custom Columns (1/4) 00:00:00:00

Generating the output list view can be a time consuming operation, depending on the number of custom columns and the number of events.

The various durations for the last import, column handling, data import and data retrieval can be seen in "Help", "Info...":



SQL Event Analyzer

```
Application version:         1.0.0
SQL Server version:          13
SQL Server:                  win2016_dev\sql2016
Session Id:                  9ab44483-d610-4d20-93f5-7f8cc36c0b00
Last import time:            00:00:00:00
Last columns handling time:  00:00:00:00
Last data retrieval time:    00:00:00:00
```

"Last import time": The time used for the last import of the chosen trace files to the database.

"Last columns handling time": The time used for the custom columns to handle the imported data.

"Last data retrieval time": The time used for retrieving the handled data from the database.

4.2 Input- and output types

The following input types are available for custom columns:

RegEx:	Regular Expression
SQL:	SQL query
Constant:	A constant value
Stored Procedure name:	Syntax: `'name', occurrence`

`name` is the name of a Stored Procedure and is the value after an "`exec`" or an "`execute`". If "`exec`" or "`execute`" is not given, the name will not be found.

`occurrence` states which Stored Procedure should be used. Normally this value will be 1, but if more than one Stored Procedure are referred in the same batch, this value can be used to state which of the Stored Procedures that should be used.

Example of an output value for Stored Procedure name: `'S_Proc1', 1`

The following output types are available for custom columns:

RegEx:	Regular Expression
SQL:	SQL query
Constant:	A constant value
Stored Procedure name:	Syntax: `occurrence`

`occurrence` states which Stored Procedure should be used. Normally this value will be 1, but if more than one Stored Procedure are referred in the same batch, this value can be used to state which of the Stored Procedures that should be returned.

Example of an output value for Stored Procedure name: 1

Stored Procedure parameter: Syntax: `position, occurrence`

`position` states a parameter position for a Stored Procedure.

Example: If the 2nd parameter (`test`) in the following should be retrieved:

```
exec S_Procedure 123, 'test', 456
```

the `position` parameter must be 2. Note, in text parameters the two single quotes that encapsulates the text parameter will not be returned.

`occurrence` states which Stored Procedure should be used. Normally this value will be 1, but if more than one Stored Procedure are referred in the same batch, this value can be used to state which of the Stored Procedures that should be used.

Example of an output value for Stored Procedure parameter:

```
'S_Proc1', 1
```

Log parameter: States a parameter to a log. The value must be a parameter position.

The Log information must be in the following format:

```
-- Log: parameter1, parameter2, ...
```

The parameters for the log information must fulfil the same requirements as the parameters for a Stored Procedure.

Example: If the 2nd parameter (`123`) int the following should be retrieved:

```
-- Log: 'test', 123
```

the value in output must be 2.

If "SQL" is used as an output type, the SQL query must fulfil the requirements for a SQL sub query. E.g. the following is possible:

```
select
case TextData
  when 'S_ViewOrder' then 'View Order'
end
```

and the following is not possible:

```
if TextData = 'S_ViewOrder'
begin
  select 'View Order'
end
```

"RegEx", "Stored Procedure name", "Stored Procedure parameter" and "Log parameter" can only be used if CLR functionality is enabled on the SQL Server. Refer to the "CLR functionality with the SQL" section for more information.

If the output type for the custom column is "SQL", there will be a limit on the number of characters the returned value contains. If the returned value has more than 50 characters, none of the values in the whole custom column will be set (not even values that are less than or equal to 50 characters).

The following SQL query for a "SQL" output type is exactly 50 characters and is valid:

```
select 12345678901234567890123456789012345678901234567890
```

The following SQL query for a "SQL" output type is 51 characters and is invalid:

```
select 123456789012345678901234567890123456789012345678901
```

If the above invalid SQL query exists, all values in the whole custom column will be empty.

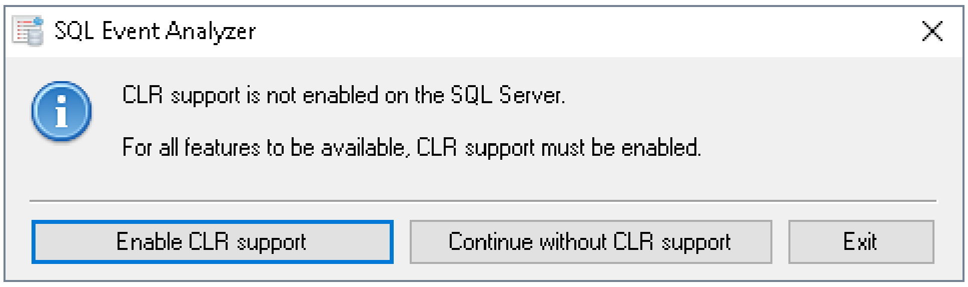
The following shows an example of how an invalid SQL query can be rewritten to only return the first 50 characters, and be valid:

```
select convert(varchar(50), 123456789012345678901234567890123456789012345678901)
```

4.3 CLR functionality

To be able to use input- and output types that are not "SQL" and "Constant", CLR functionality must be enabled on the SQL Server.

By default, CLR functionality is not enabled on a SQL Server. If CLR functionality is not enabled, SQL Event Analyzer will give an option to enable it upon start:


SQL Event Analyzer

CLR support is not enabled on the SQL Server.

For all features to be available, CLR support must be enabled.

It is possible to start SQL Event Analyzer without CLR functionality. If doing so, the following input- and output types will not be available:

- RegEx
- Stored Procedure name
- Stored Procedure parameter
- Log parameter

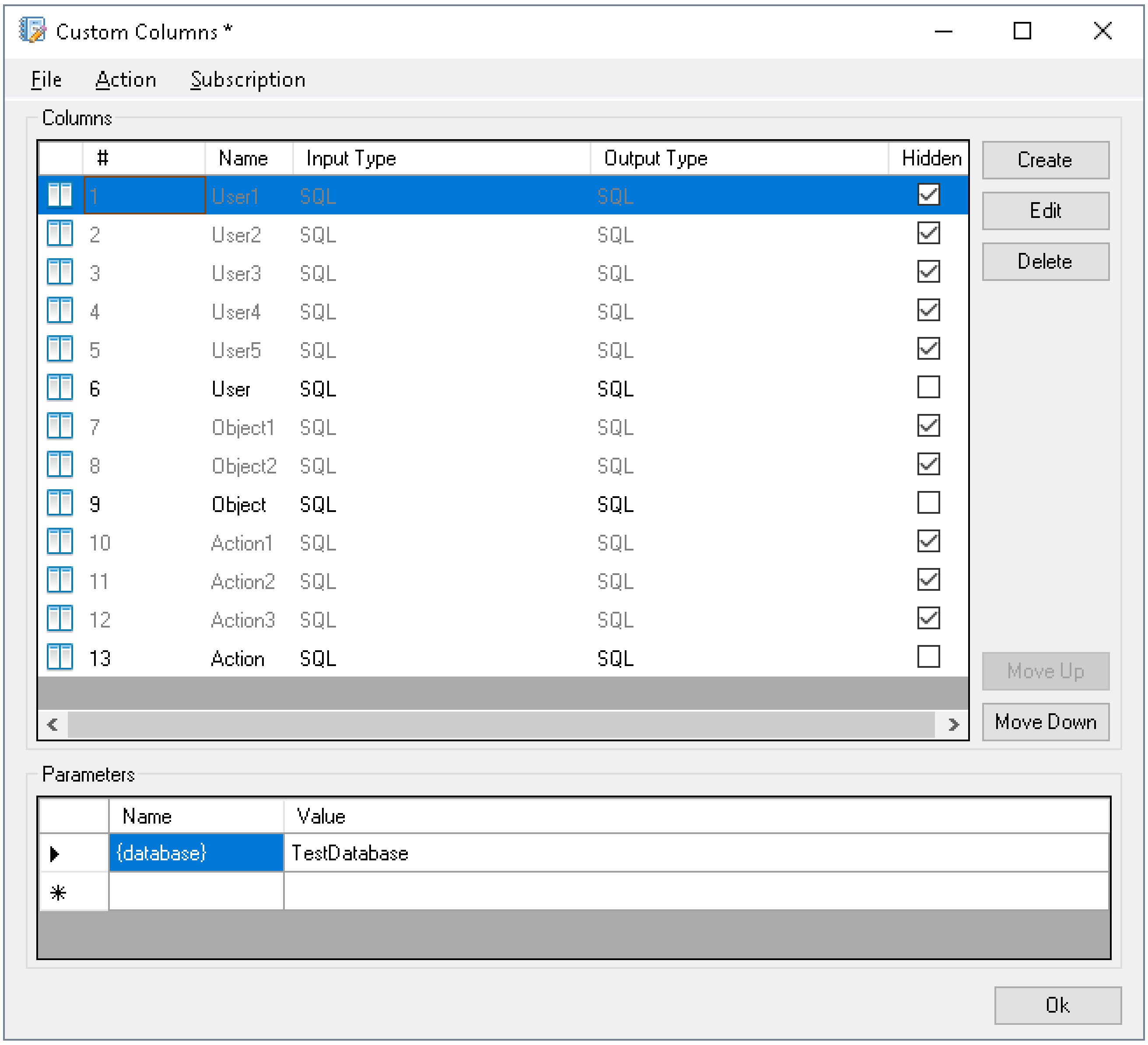
It is possible to enable temporary CLR functionality by holding down the Shift key while choosing "Enable CLR support". If temporary CLR functionality is enabled, CLR functionality will be disabled again when SQL Event Analyzer closes or a connection is made to another SQL Server.

Note, if temporary CLR functionality is enabled and SQL Event Analyzer is not properly closed (e.g. power outage), CLR functionality will not be disabled again.

4.4 Working with custom columns

Custom columns can refer each other. A reference to another custom column is only possible if the referred custom column has a lower number than the current column.

Example:

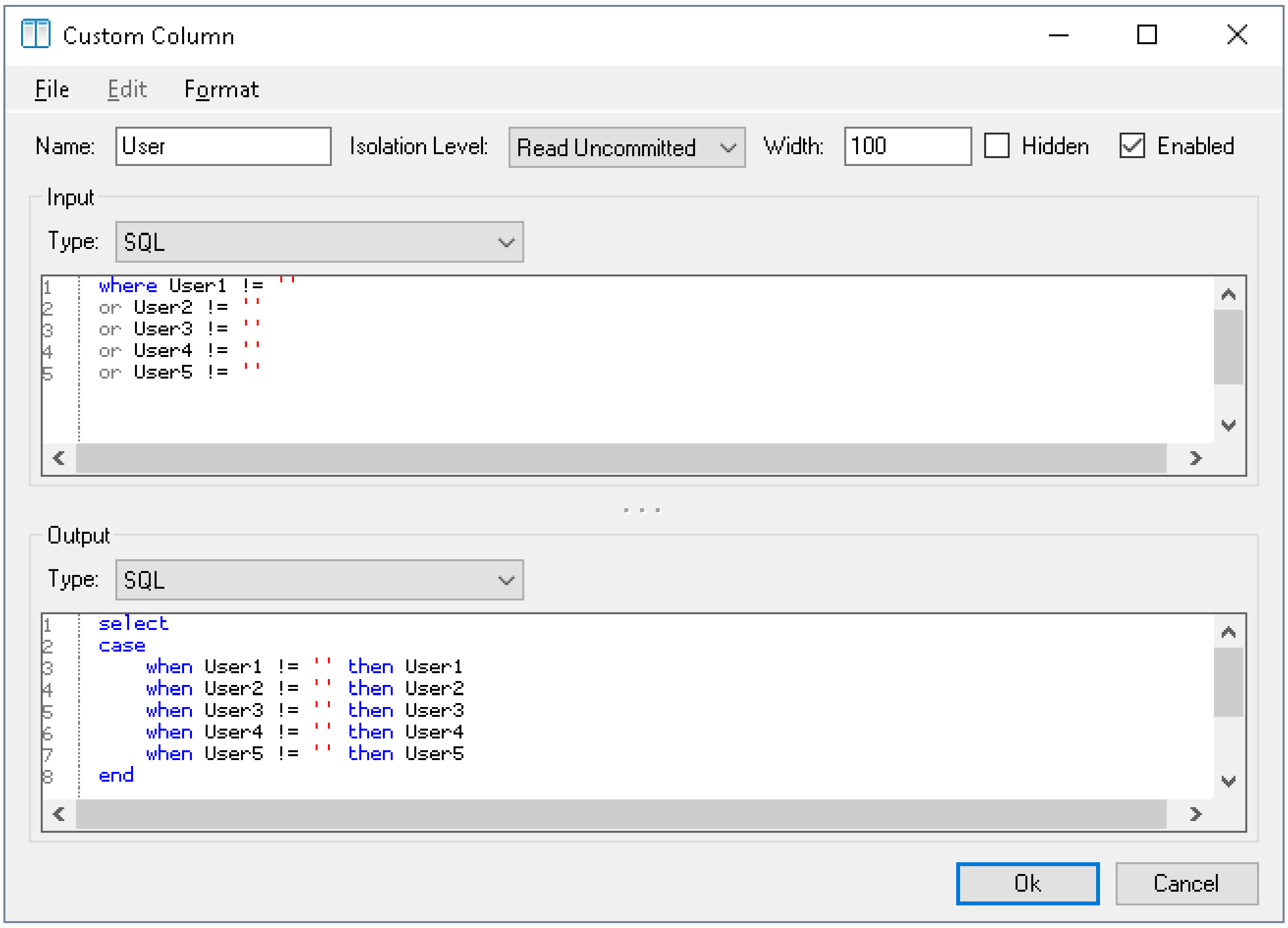


#	Name	Input Type	Output Type	Hidden
1	User1	SQL	SQL	✓
2	User2	SQL	SQL	✓
3	User3	SQL	SQL	✓
4	User4	SQL	SQL	✓
5	User5	SQL	SQL	✓
6	User	SQL	SQL	☐
7	Object1	SQL	SQL	✓
8	Object2	SQL	SQL	✓
9	Object	SQL	SQL	☐
10	Action1	SQL	SQL	✓
11	Action2	SQL	SQL	✓
12	Action3	SQL	SQL	✓
13	Action	SQL	SQL	☐

Name	Value
{database}	TestDatabase

In the above, column number 6 (User) can refer the columns from number 1 to 5, but not refer the columns from 7 to 13.

The following shows column 6 (User) referring column number 1 to 5:



Name: User

Isolation Level: Read Uncommitted

Width: 100

Hidden [✓ Enabled]

Input

Type: SQL

```
where User1 != ''
or User2 != ''
or User3 != ''
or User4 != ''
or User5 != ''
```

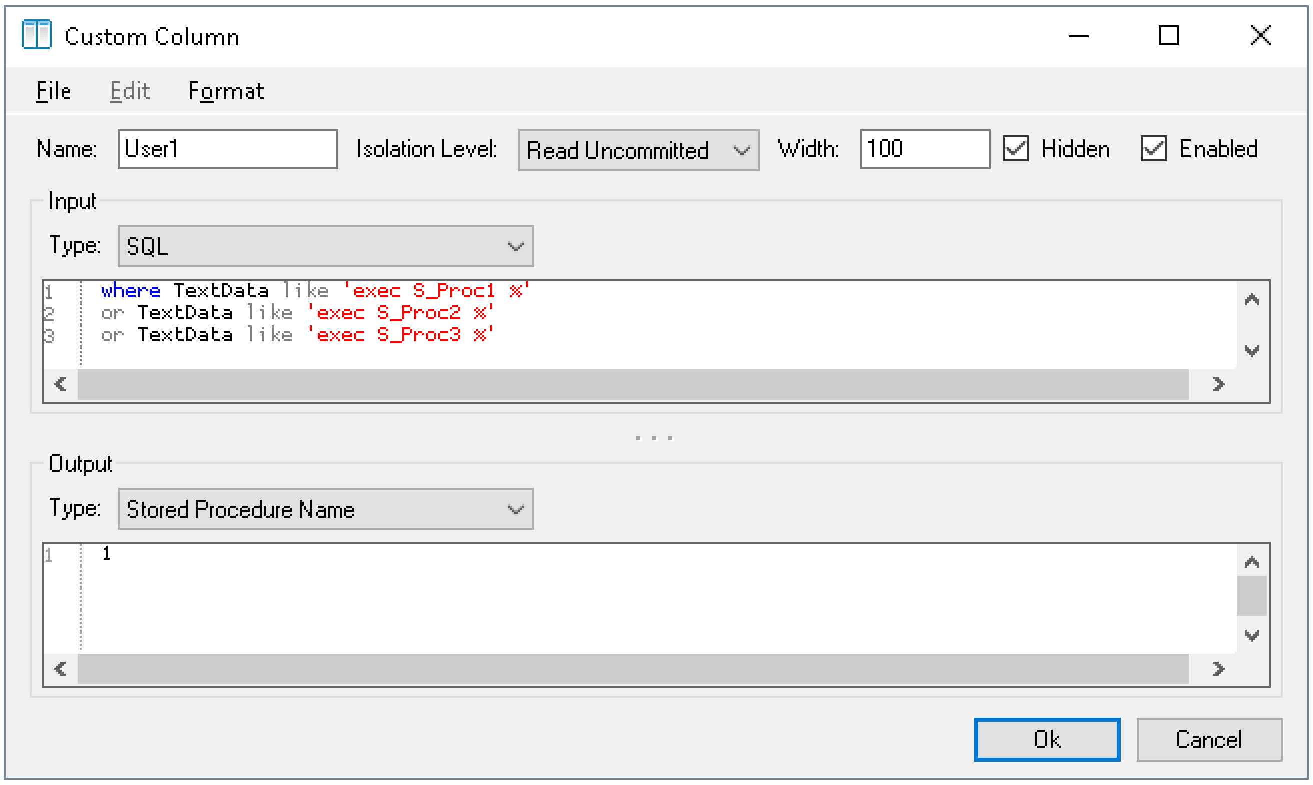
Output

Type: SQL

```
select
case
    when User1 != '' then User1
    when User2 != '' then User2
    when User3 != '' then User3
    when User4 != '' then User4
    when User5 != '' then User5
end
```

A custom column can always refer a standard column.

The following shows column 1 (User1) refer the standard column "TextData":



Name: User1
Isolation Level: Read Uncommitted
Width: 100
Hidden ✓
Enabled ✓

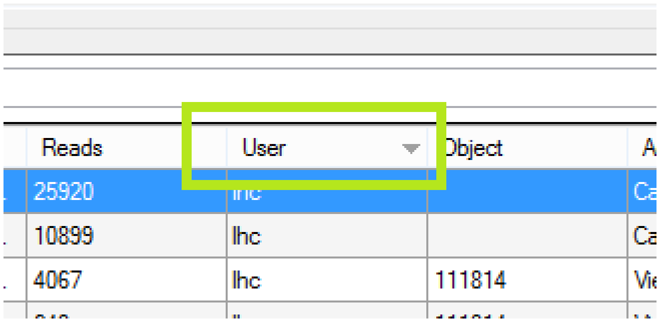
Input
Type: SQL

```
where TextData like 'exec S_Proc1 %'
or TextData like 'exec S_Proc2 %'
or TextData like 'exec S_Proc3 %'
```

Output
Type: Stored Procedure Name

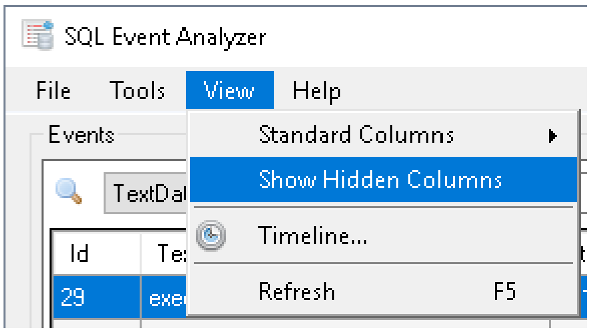
```
1
```

In the above examples are column 6 (User) shown in the output list view and column 1 (User1) is hidden:

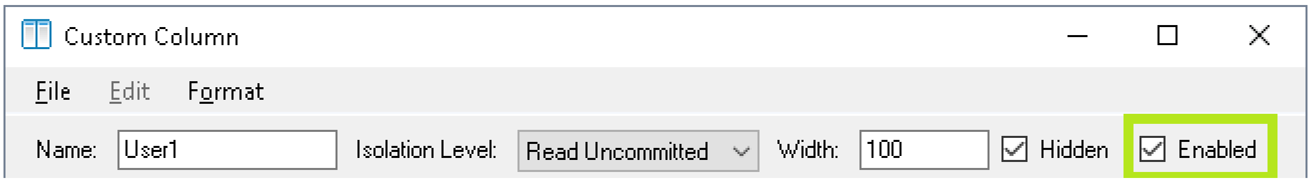


Reads	User	Object	A
25920	lhc		Ca
10899	lhc		Ca
4067	lhc	111814	Vie

It is possible to show hidden columns in the output list view by choosing "View", "Show Hidden Columns":



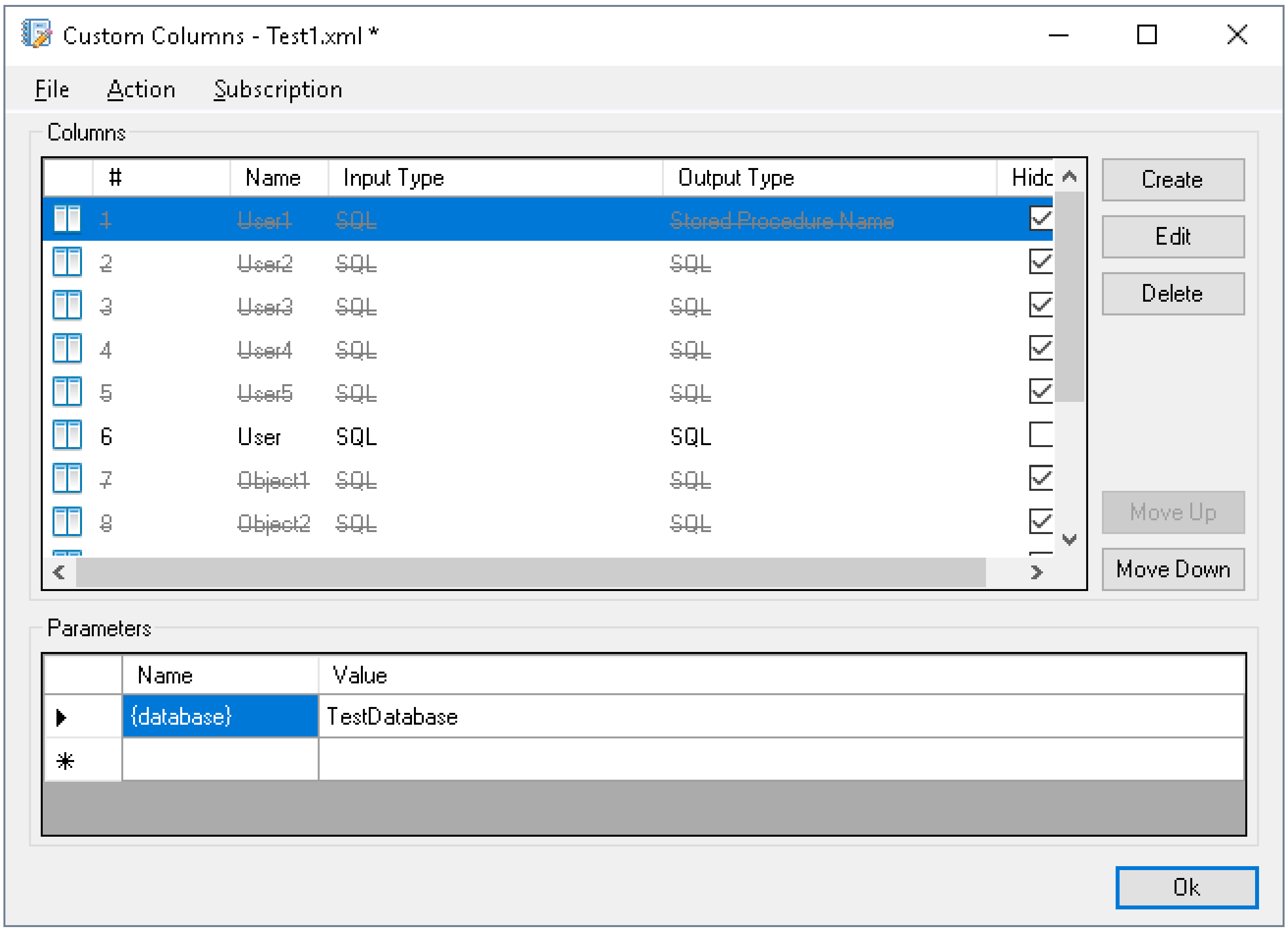
A custom column can be marked a active or inactive:



Name: User1 Isolation Level: Read Uncommitted Width: 100 [✓ Hidden] [✓ Enabled]

If a column is inactive, it is not possible to refer it from another custom column, and it will not be included in the output list view.

Inactive columns are shown with the column text strikethrough:



#	Name	Input Type	Output Type	Hidc
~~1~~	~~User1~~	~~SQL~~	~~Stored Procedure Name~~	✓
~~2~~	~~User2~~	~~SQL~~	~~SQL~~	✓
~~3~~	~~User3~~	~~SQL~~	~~SQL~~	✓
~~4~~	~~User4~~	~~SQL~~	~~SQL~~	✓
~~5~~	~~User5~~	~~SQL~~	~~SQL~~	✓
6	User	SQL	SQL	☐
~~7~~	~~Object1~~	~~SQL~~	~~SQL~~	✓
~~8~~	~~Object2~~	~~SQL~~	~~SQL~~	✓

Name	Value
{database}	TestDatabase

If CLR functionality is not enabled on the SQL Server, columns that are using CLR functionality will be inactive, and it will not be possible to activate these columns.

4.5 CLR functionality with the SQL type

The following functions can be used if the input- or output type are "SQL":

- `GetRegEx`: Returns RegEx group 1, that matches the given RegEx
- `MatchRegEx`: Returns a boolean value indicating if the given RegEx can be matched
- `GetStoredProcedureName`: Returns the name of a Stored Procedure found in a search text
- `MatchStoredProcedureName`: Returns a boolean value indicating if the name of a Stored Procedure is found in a search text
- `GetStoredProcedureParameter`: If a parameter is found on the given position it is returned
- `MatchStoredProcedureParameter`: Returns a boolean value indicating if a Stored Procedure in a search text contains a given value on a given position
- `GetLogParameter`: If a parameter is found on the given position it is returned
- `MatchLogParameter`: Returns a boolean value indicating if a log in a search text contains a given value on a given position

Input- and output values for the above CLR functions:

- `string GetRegEx(string input, string regExPattern)`
- `bool MatchRegEx(string input, string regExPattern)`
- `string GetStoredProcedureName(string input, int occurrence)`
- `bool MatchStoredProcedureName(string input, string name, int occurrence)`
- `string GetStoredProcedureParameter(string input, int position, int occurrence)`
- `bool MatchStoredProcedureParameter(string input, int position, string value, int occurrence)`
- `string GetLogParameter(string input, int position)`
- `bool MatchLogParameter(string input, int position, string value)`

The following examples shows various usages of the above CLR functions. It is wanted to return all events from a TraceData table, where the first parameter to a Stored Procedure must be "lhc". Furthermore is it wanted that the first column in the result set is the actual name of the found Stored Procedure:

```
select dbo.GetStoredProcedureName(t.TextData, 1), t.*
from dbo.TraceData_f63631c1-5824-4dce-b631-f45e187c3cc1 t
where dbo.GetRegEx(t.TextData, 'exec S_Get.+ ''(.+?)''') = 'lhc'

select dbo.GetStoredProcedureName(t.TextData, 1), t.*
from TraceData_f63631c1-5824-4dce-b631-f45e187c3cc1 t
where dbo.MatchStoredProcedureParameter(t.TextData, 1, 'lhc', 1) = 1

select dbo.GetStoredProcedureName(t.TextData, 1), t.*
from TraceData_f63631c1-5824-4dce-b631-f45e187c3cc1 t
where dbo.GetStoredProcedureParameter(t.TextData, 1, 1) = 'lhc'
```

If `GetStoredProcedureParameter`, `MatchStoredProcedureParameter`, `GetLogParameter` or `MatchLogParameter` is used with 0 as the `position` parameter, the following will be returned:

- `GetStoredProcedureParameter`: Number of parameters
- `MatchStoredProcedureParameter`: Boolean value indicating if the number of parameters are equal to value
- `GetLogParameter`: Number of parameters
- `MatchLogParameter`: Boolean value indicating if the number of parameters are equal to value

For `GetStoredProcedureName`, `MatchStoredProcedureName`, `GetStoredProcedureParameter` and `MatchStoredProcedureParameter`, an `occurrence` parameter must be given. This value states which Stored Procedure that should be used if more than one Stored Procedure exists in the same batch. Example, by executing the following two Stored Procedures in the same batch:

```
exec s_proc1
exec s_proc2
```

the returned value will be `s_proc1` if the `occurrence` parameter is 1, and `s_proc2` if the `occurrence` parameter is 2.

4.6 TextData Cleaned

The same SQL statement can be executed in different ways. E.g. the following two Stored Procedure executions are the same:

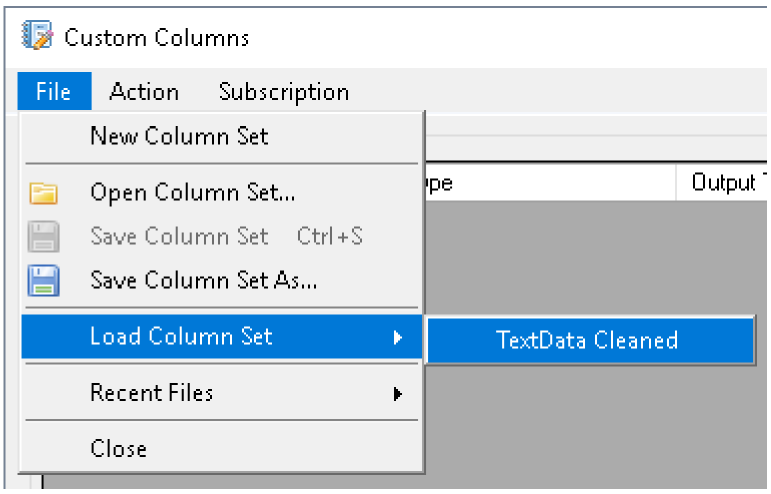
```
exec s_proc1
execute [s_proc1]
```

SQL Event Analyzer contains functionality to "clean" the TextData so it appears as unified as possible. After cleaning the two above Stored Procedure executions, both of them will be presented as:

```
s_proc1
```

This makes it possible to e.g. view statistics for how many times a SQL query has been executed.

The TextData cleaning functionality can be used by choosing "Load Column Set", "TextData Cleaned" in the "Custom Columns" editor:



4.7 Example of index creation on a custom column

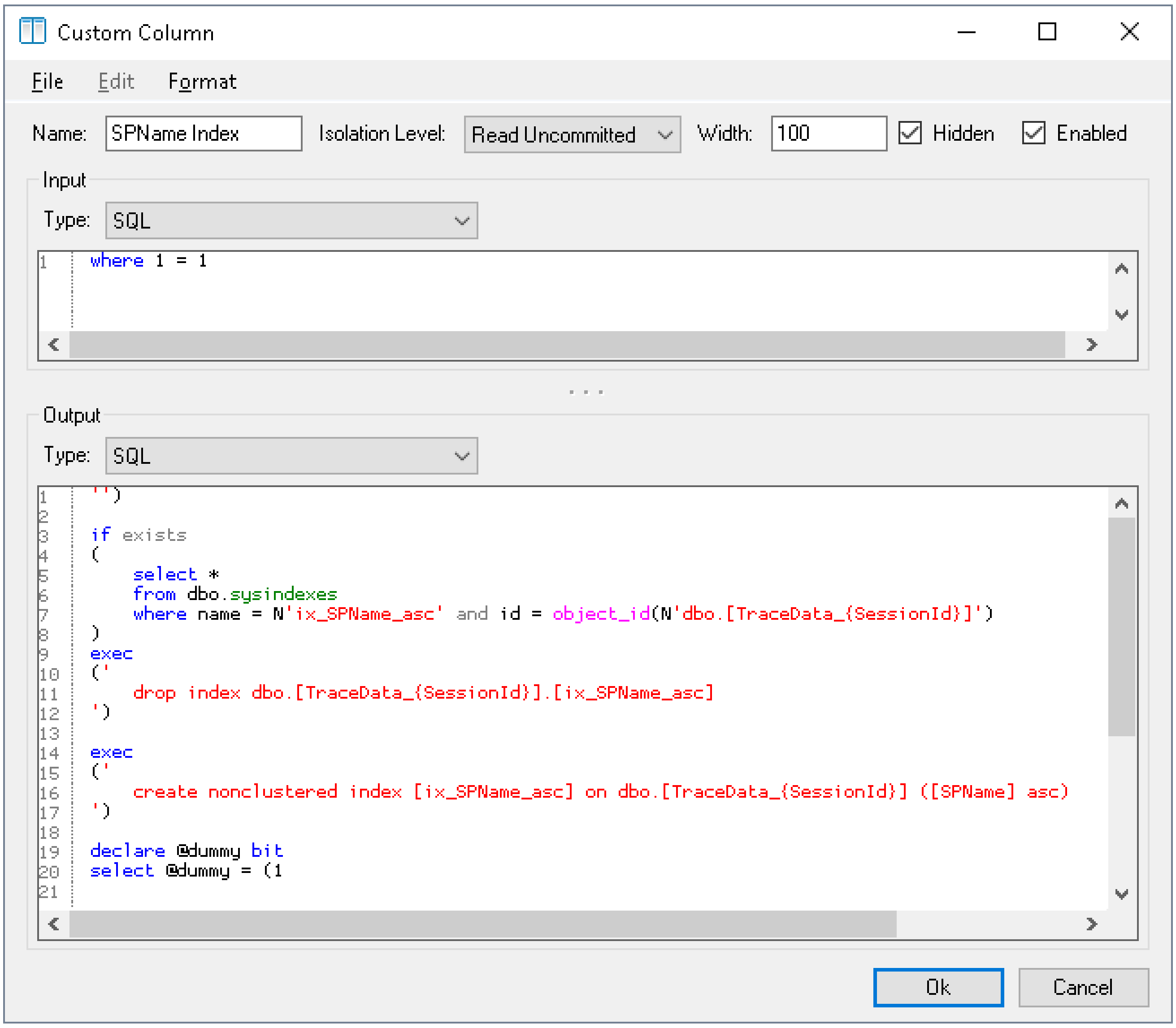
It is possible to operate on either DML or DDL from the custom columns. This is possible by breaking the functionality in the SQL output type:

Breaking the SQL output type can be done by inserting the following:

```
'')

...

declare @dummy bit
select @dummy = (1
```

Where ... contains the SQL statement that is wanted to be executed.

The following shows an example of creating an index on a custom column with the name SPName:



Name: SPName Index Isolation Level: Read Uncommitted Width: 100 [✓ Hidden] [✓ Enabled]

Input

Type: SQL

```
where 1 = 1
```

Output

Type: SQL

```
'')

if exists
(
    select *
    from dbo.sysindexes
    where name = N'ix_SPName_asc' and id = object_id(N'dbo.[TraceData_{SessionId}]')
)
exec
('
    drop index dbo.[TraceData_{SessionId}].[ix_SPName_asc]
')

exec
('
    create nonclustered index [ix_SPName_asc] on dbo.[TraceData_{SessionId}] ([SPName] asc)
')

declare @dummy bit
select @dummy = (1
```

Note, in the above the parameter `{SessionId}` is used. This parameter will be replaced with the correct session id when handling the custom columns.

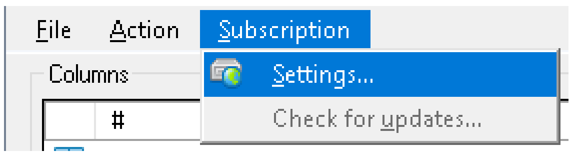
The SQL statement in the above screenshot is:

```
'')

if exists
(
            select *
            from dbo.sysindexes
            where name = N'ix_SPName_asc' and id = 
object_id(N'dbo.[TraceData_{SessionId}]')
)
exec
('
            drop index dbo.[TraceData_{SessionId}].[ix_SPName_asc]
')

exec
('
            create nonclustered index [ix_SPName_asc] on dbo.[TraceData_{SessionId}]
([SPName] asc)
')

declare @dummy bit
select @dummy = (1
```

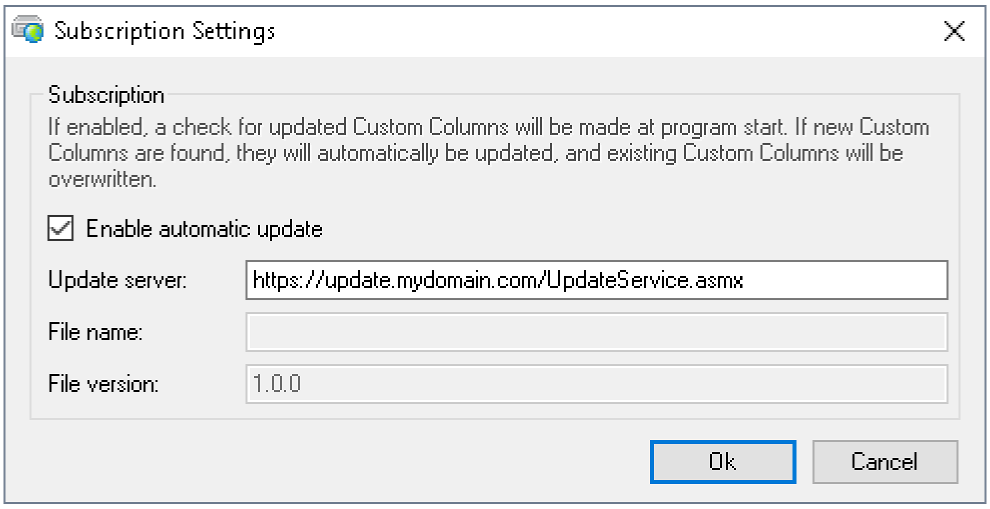
4.8 Subscription

It is possible to subscribe to custom columns, and automatically download new versions of a custom columns set from a centralized server.

The subscription functionality can be activated in "Subscription", "Settings..." in the "Custom Columns" editor:



In the settings, the server can be entered and it can be chosen to activate or deactivate the functionality:


Subscription Settings

Subscription

If enabled, a check for updated Custom Columns will be made at program start. If new Custom Columns are found, they will automatically be updated, and existing Custom Columns will be overwritten.

[✓] Enable automatic update

Update server: https://update.mydomain.com/UpdateService.asmx

File name:

File version: 1.0.0

The subscription functionality is attached to the active custom columns set, i.e. it is possible to define different servers for different custom columns sets.

Signature for the web service communication:

```
public string GetDownloadUrl(string productName, string machineName, string userName, string
domainName, string allowUpdateCriteria)
```

```
public string GetLatestVersion(string productName, string currentVersion, string
machineName, string userName, string domainName, string allowUpdateCriteria)
```

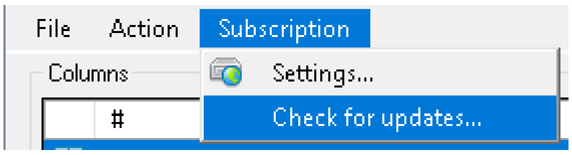
```
public string GetChangelogUrl(string productName, string allowUpdateCriteria)
```

Changelog format is:

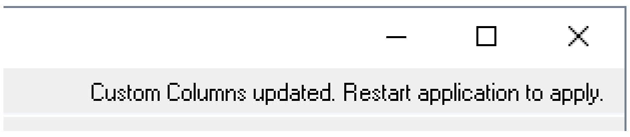
```
1.0.1

Feature X implemented.


1.0.0

Initial version.
```

When the subscription functionality is activated, a check for a newer version will be performed upon program start and when loading a new custom columns file.

It is also possible to perform a manual update check from "Subscription", "Check for updates...":



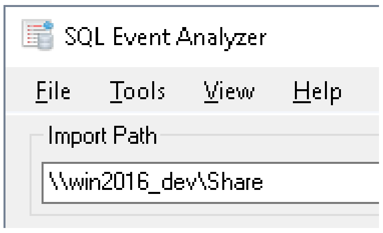
If an update is found for the custom columns set, it will be shown in the user interface:


Custom Columns updated. Restart application to apply.

The new custom columns will not be active until the application has been restarted.

5. Events from trace files

Trace files can be imported from the path given in "Import Path":



\\win2016_dev\Share

For a trace file to be able to be imported in SQL Event Analyzer, it must contain a least the following data:

- TextData
- EventClass
- StartTime

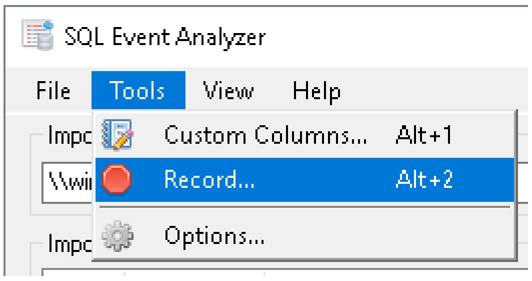
E.g. a trace file from SQL Server Profiler can be imported.

If SQL Server 2012 or newer is used, then it is also possible to import Extended Event Trace files (.xel).

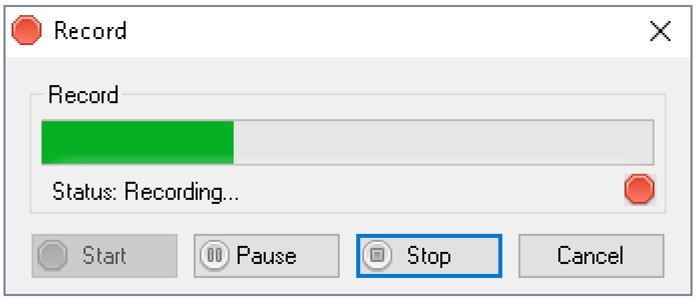
6. Real time events

Instead of importing events from a trace file, the events can be recorded in real time.

To start a recording, choose "Tools", "Record...":



When recording, all events on the SQL Server will be registered.

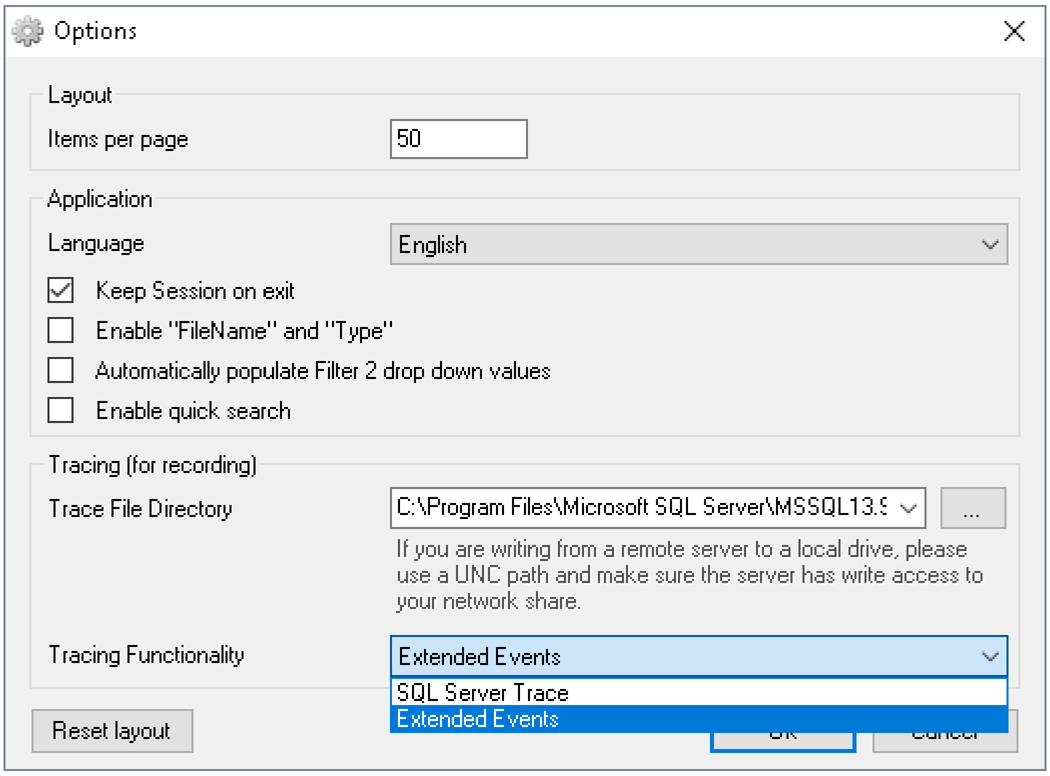


Status: Recording...

6.1 Changing tracing functionality

If using SQL Server 2012 or newer, it is possible to change the tracing functionality to be used when recording real time events.

If SQL Event Analyzer is connected to a SQL Server 2012 or newer, the tracing functionality will by default be set to "Extended Events". This can be changed in the "Options" menu:



Options

Layout

Items per page: 50

Application

Language: English

- [✓] Keep Session on exit
- [ ] Enable "FileName" and "Type"
- [ ] Automatically populate Filter 2 drop down values
- [ ] Enable quick search

Tracing (for recording)

Trace File Directory: C:\Program Files\Microsoft SQL Server\MSSQL13.S

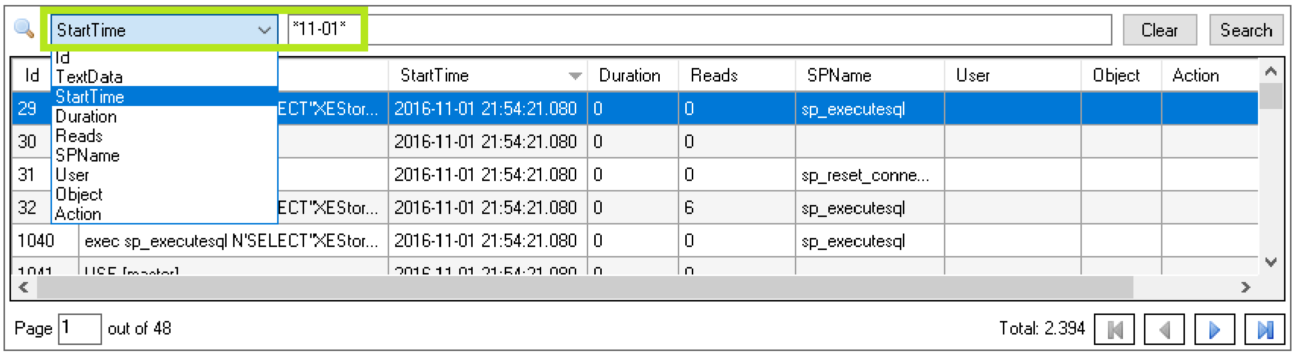
If you are writing from a remote server to a local drive, please use a UNC path and make sure the server has write access to your network share.

Tracing Functionality: Extended Events

Note, it is not possible to choose "Extended Events" if using a SQL Server version older than SQL Server 2012.

7. Searching and sorting

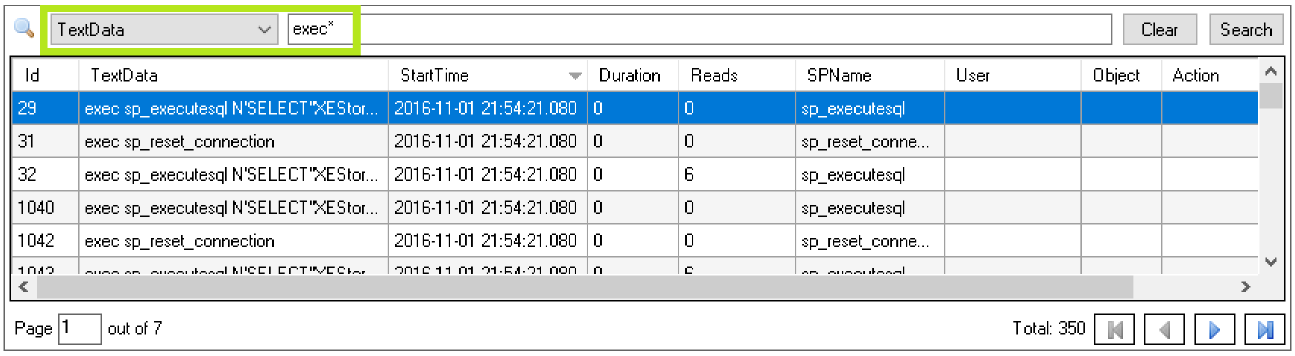
It is possible to search in all columns. Search supports wild cards by using *. All columns except from the "TextData" column supports the use of wildcards before- and after the search term:



Id	TextData	StartTime	Duration	Reads	SPName	User	Object	Action
29	ECT"XEStor...	2016-11-01 21:54:21.080	0	0	sp_executesql			
30		2016-11-01 21:54:21.080	0	0				
31		2016-11-01 21:54:21.080	0	0	sp_reset_conne...			
32	ECT"XEStor...	2016-11-01 21:54:21.080	0	6	sp_executesql			
1040	exec sp_executesql N'SELECT"XEStor...	2016-11-01 21:54:21.080	0	0	sp_executesql			
1041	USE [master]	2016-11-01 21:54:21.080	0	0				

Page 1 out of 48 Total: 2.394

When searching in the "TextData" column, wildcards are only supported after the search term:

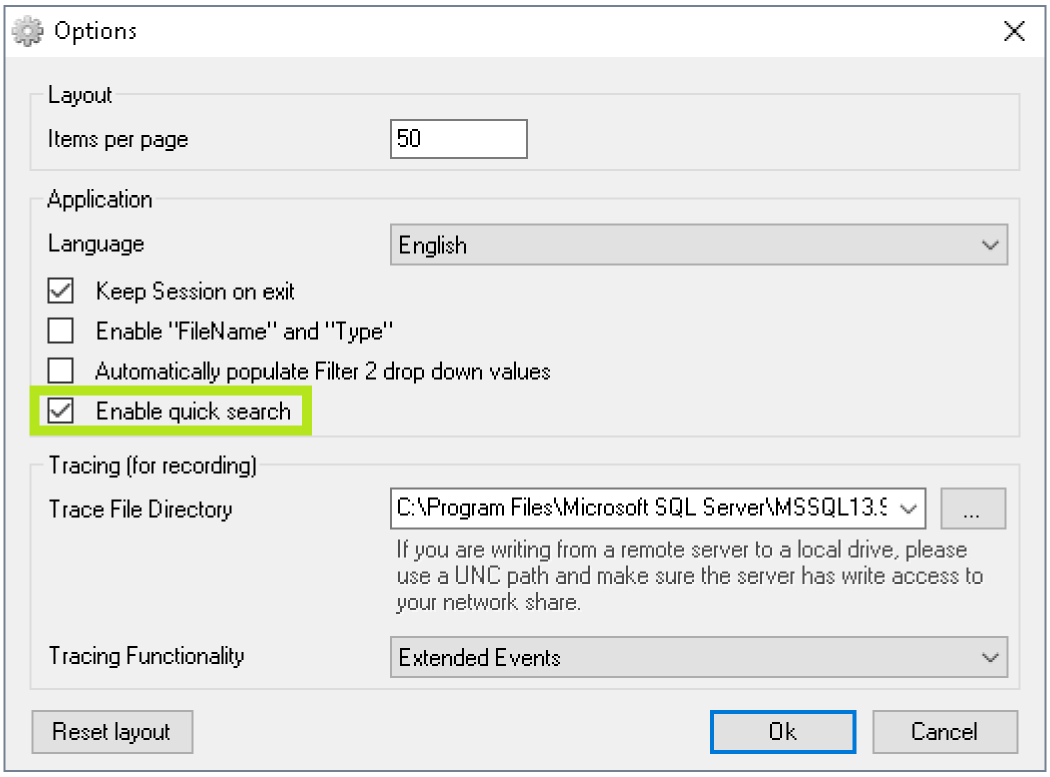


Id	TextData	StartTime	Duration	Reads	SPName	User	Object	Action
29	exec sp_executesql N'SELECT"XEStor...	2016-11-01 21:54:21.080	0	0	sp_executesql			
31	exec sp_reset_connection	2016-11-01 21:54:21.080	0	0	sp_reset_conne...			
32	exec sp_executesql N'SELECT"XEStor...	2016-11-01 21:54:21.080	0	6	sp_executesql			
1040	exec sp_executesql N'SELECT"XEStor...	2016-11-01 21:54:21.080	0	0	sp_executesql			
1042	exec sp_reset_connection	2016-11-01 21:54:21.080	0	0	sp_reset_conne...			
1043	exec sp_executesql N'SELECT"XEStor...	2016-11-01 21:54:21.080	0	6	sp_executesql			

Page 1 out of 7 Total: 350

All columns can be sorted, so it is possible to e.g. see the event with the longest duration.

Note, the shown searches are only possible if "Enable quick search" is enabled in "Options":


Options

Layout

Items per page: 50

Application

Language: English

- [✓] Keep Session on exit
- [ ] Enable "FileName" and "Type"
- [ ] Automatically populate Filter 2 drop down values
- [✓] Enable quick search

Tracing (for recording)

Trace File Directory: C:\Program Files\Microsoft SQL Server\MSSQL13.S

If you are writing from a remote server to a local drive, please use a UNC path and make sure the server has write access to your network share.

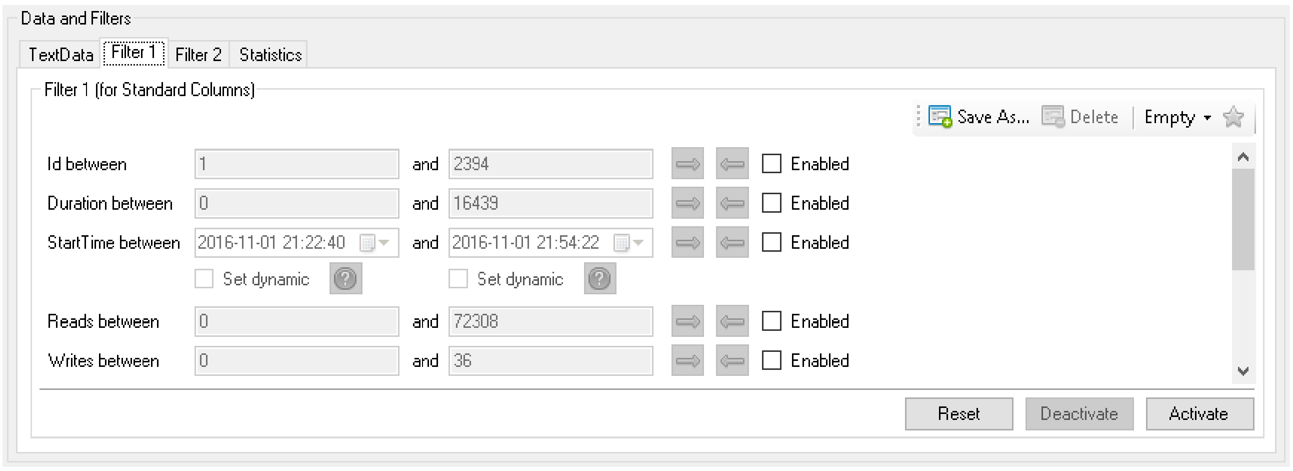
Tracing Functionality: Extended Events

If "Enable quick search" is not enabled, searches are performed in the filtering section instead. See the "Filters" section for more information.

8. Filters

The events can be filtered by using two filters:

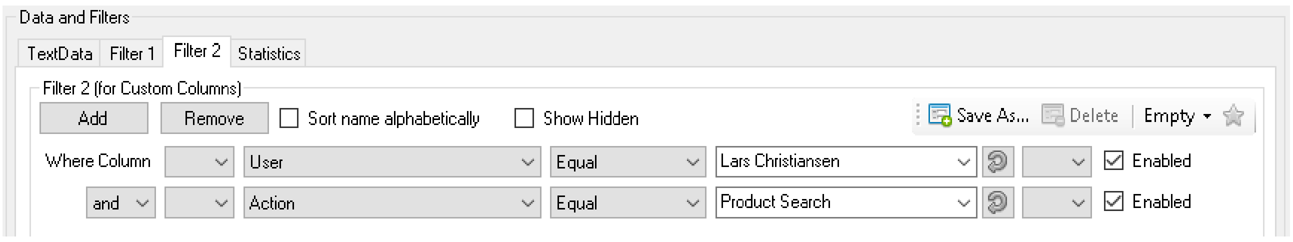
Filter 1: Filtering data from standard columns:


Data and Filters

Filter 1 (for Standard Columns)

Id between	1	and	2394	[ ] Enabled
Duration between	0	and	16439	[ ] Enabled
StartTime between	2016-11-01 21:22:40	and	2016-11-01 21:54:22	[ ] Enabled
	[ ] Set dynamic		[ ] Set dynamic	
Reads between	0	and	72308	[ ] Enabled
Writes between	0	and	36	[ ] Enabled

Filter 2: Filtering data from custom columns:


Data and Filters

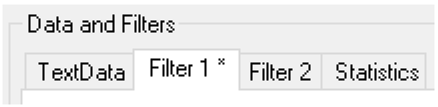
TextData | Filter 1 | Filter 2 | Statistics

Filter 2 (for Custom Columns)

Add | Remove | ☐ Sort name alphabetically | ☐ Show Hidden | Save As... | Delete | Empty

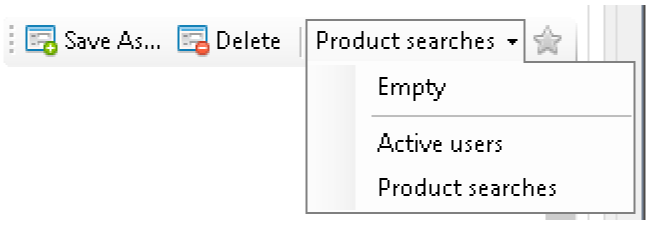
Where Column		User	Equal	Lars Christiansen		☑ Enabled
and		Action	Equal	Product Search		☑ Enabled

When a filter is active, the name of the filter will be marked with a *:


Data and Filters

TextData | Filter 1 * | Filter 2 | Statistics

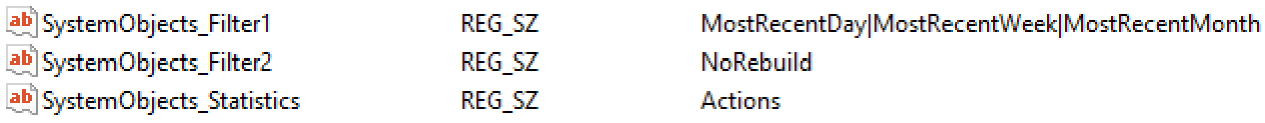
Searches defined in Filter 1 or Filter 2 can be saved so they can be reused. To save a search, choose "Save As..." and enter a name. Then the saved search will be shown in the list as shown in the following:



Likewise, the searches can be deleted or modified.

Saved searched can be marked as system objects. If a saved search is marked as a system object, then it is not possible to delete, modify or overwrite the search.

Saved searched can be marked as system objects by adding the name of the saved searches to the registry key for SQL Event Analyzer. The name for the saved searched must be separated by | like:



SystemObjects_Filter1	REG_SZ	MostRecentDay|MostRecentWeek|MostRecentMonth
SystemObjects_Filter2	REG_SZ	NoRebuild
SystemObjects_Statistics	REG_SZ	Actions

The name for the saved search can not contain any of the following characters:

, ' & [ ]

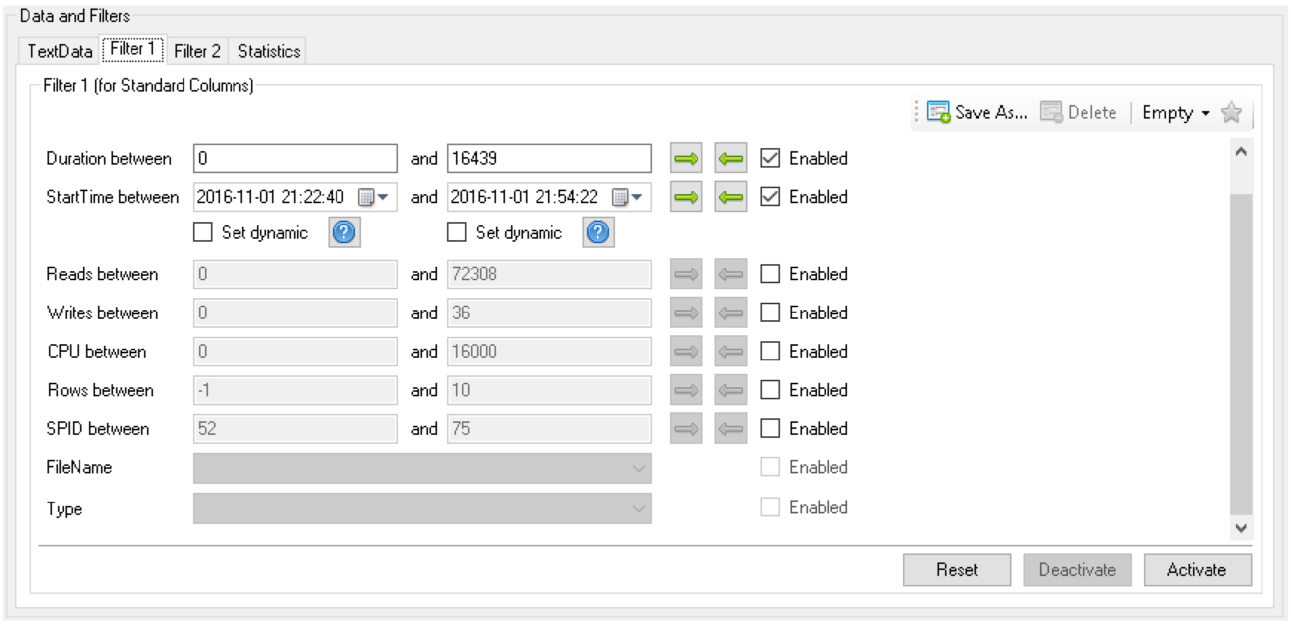
Likewise, the name can not be "Empty" or contain any character that is not allowed in a file name.

8.1 Filter 1

When starting SQL Event Analyzer, the values for Filter 1 will be prefilled with the minimum, maximum and unique values for data. E.g. the first "StartTime" value will be prefilled with the time for the first of the events, and the second "StartTime" will be prefilled with the time for the last of the events.

Likewise, the "Type" will be prefilled with unique values for "Type" among the events.

By default, "FileName" and "Type" are deactivated:



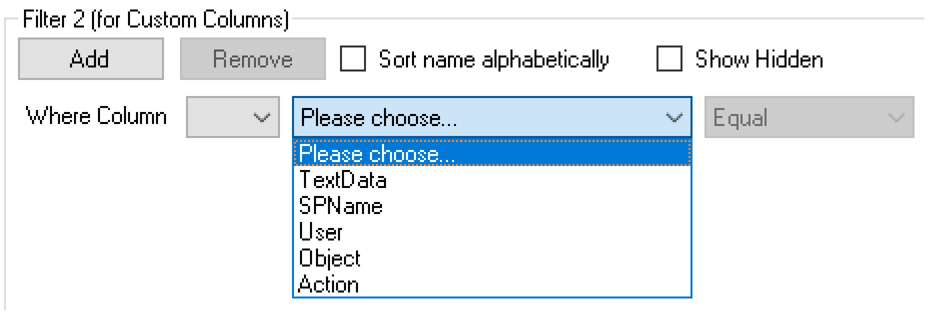
Duration between	0	and	16439	☑ Enabled
StartTime between	2016-11-01 21:22:40	and	2016-11-01 21:54:22	☑ Enabled
	☐ Set dynamic		☐ Set dynamic	
Reads between	0	and	72308	☐ Enabled
Writes between	0	and	36	☐ Enabled
CPU between	0	and	16000	☐ Enabled
Rows between	-1	and	10	☐ Enabled
SPID between	52	and	75	☐ Enabled
FileName				☐ Enabled
Type				☐ Enabled

"FileName" and "Type" can be activated in "Options". Note that if "FileName" and "Type" is activated, the handling of the TraceData tables will take more time. If "FileName" and "Type" is not activated, the values will still be saved in the TraceData tables.

By clicking on the "Reset" button, the values will be set to their initial prefilled values.

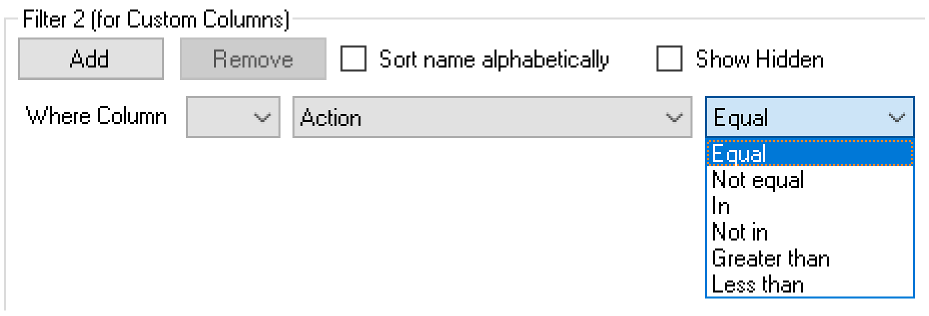
8.2 Filter 2

It is possible to add filters to custom columns. First choose the custom column:

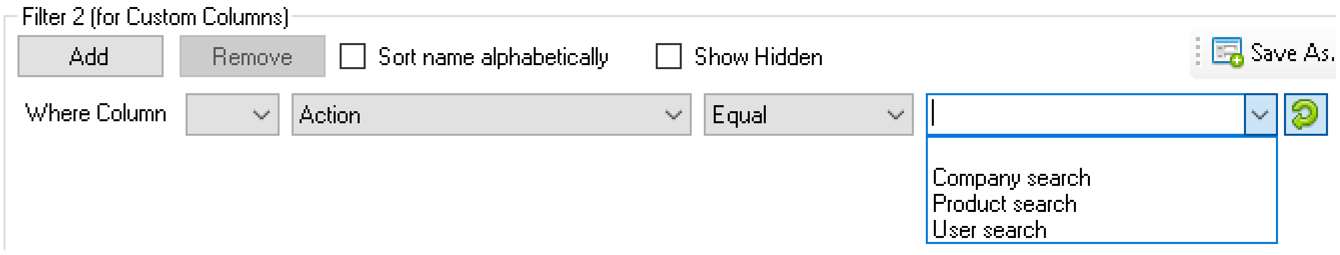


Note, "TextData" will always be at the top of the list and is not influenced by "Sort name alphabetically".

Choose criteria:

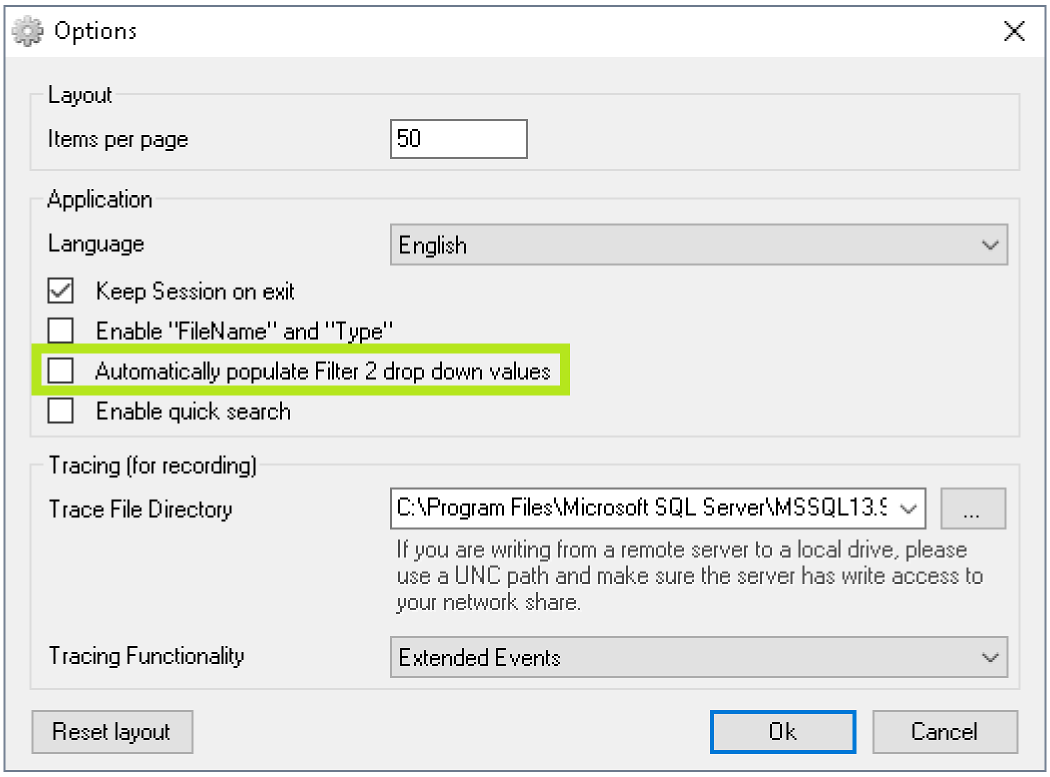


Value is chosen or entered:



It is possible to manually enter a value in the value list or choose one of the unique values for the custom column. The unique values are always shown alphabetically.

Note, by default the unique values are not set to be automatically populated. This can be changed in "Options":


Options

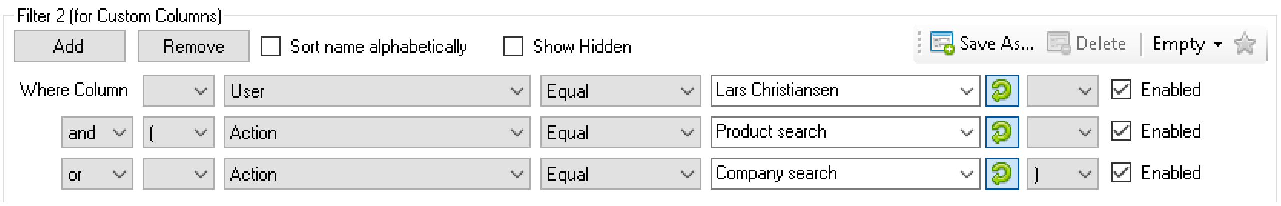
Layout
Items per page: 50

Application
Language: English
- [x] Keep Session on exit
- [ ] Enable "FileName" and "Type"
- [ ] Automatically populate Filter 2 drop down values
- [ ] Enable quick search

Tracing (for recording)
Trace File Directory: C:\Program Files\Microsoft SQL Server\MSSQL13.S
If you are writing from a remote server to a local drive, please use a UNC path and make sure the server has write access to your network share.
Tracing Functionality: Extended Events

Reset layout | Ok | Cancel

It is possible to add as many search criteria as wanted:

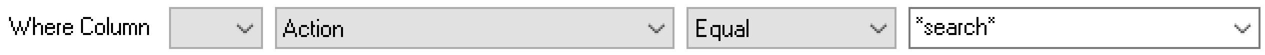

Filter 2 (for Custom Columns)

Add | Remove | [ ] Sort name alphabetically | [ ] Show Hidden | Save As... | Delete | Empty

Where Column		User	Equal	Lars Christiansen			[x] Enabled
and	(	Action	Equal	Product search			[x] Enabled
or		Action	Equal	Company search		)	[x] Enabled

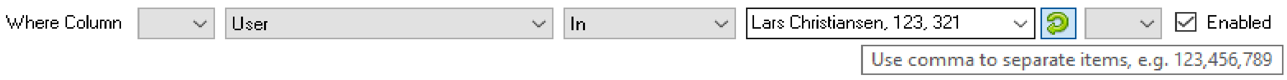
Note the use of parentheses in the above.

By using the "Equal" and "Not equal" operators, wildcards can be used. Wildcards can be used both before- and after the search term:


Where Column | | Action | Equal | *search*

The following characters can be used as wildcard characters: "%" and "*".

When using the "In" or "Not in" operators, it is possible to search for a list of values. The values must be separated by comma:


Where Column | | User | In | Lars Christiansen, 123, 321 | | [x] Enabled

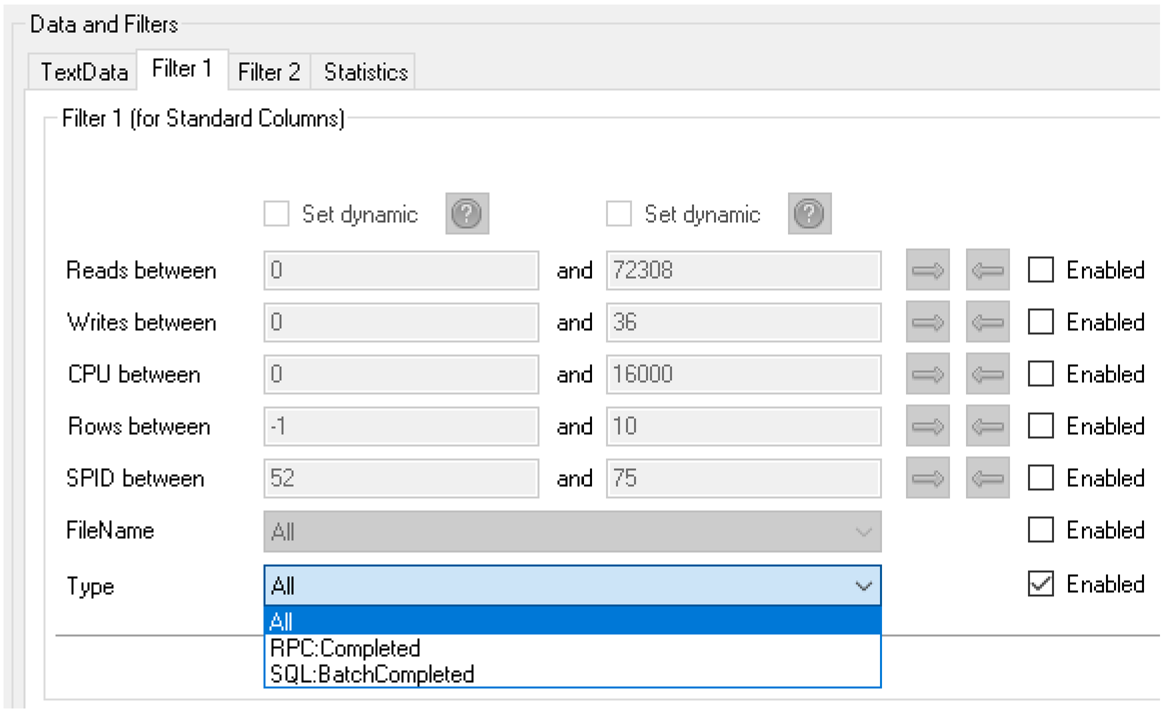
Use comma to separate items, e.g. 123,456,789

The "Active" option defines if the corresponding filter is used or not.

8.3 Filtering example

For a set of events, it is wanted to find all events where the "Type" is either "RPC:Completed" or "SQL:BatchCompleted".

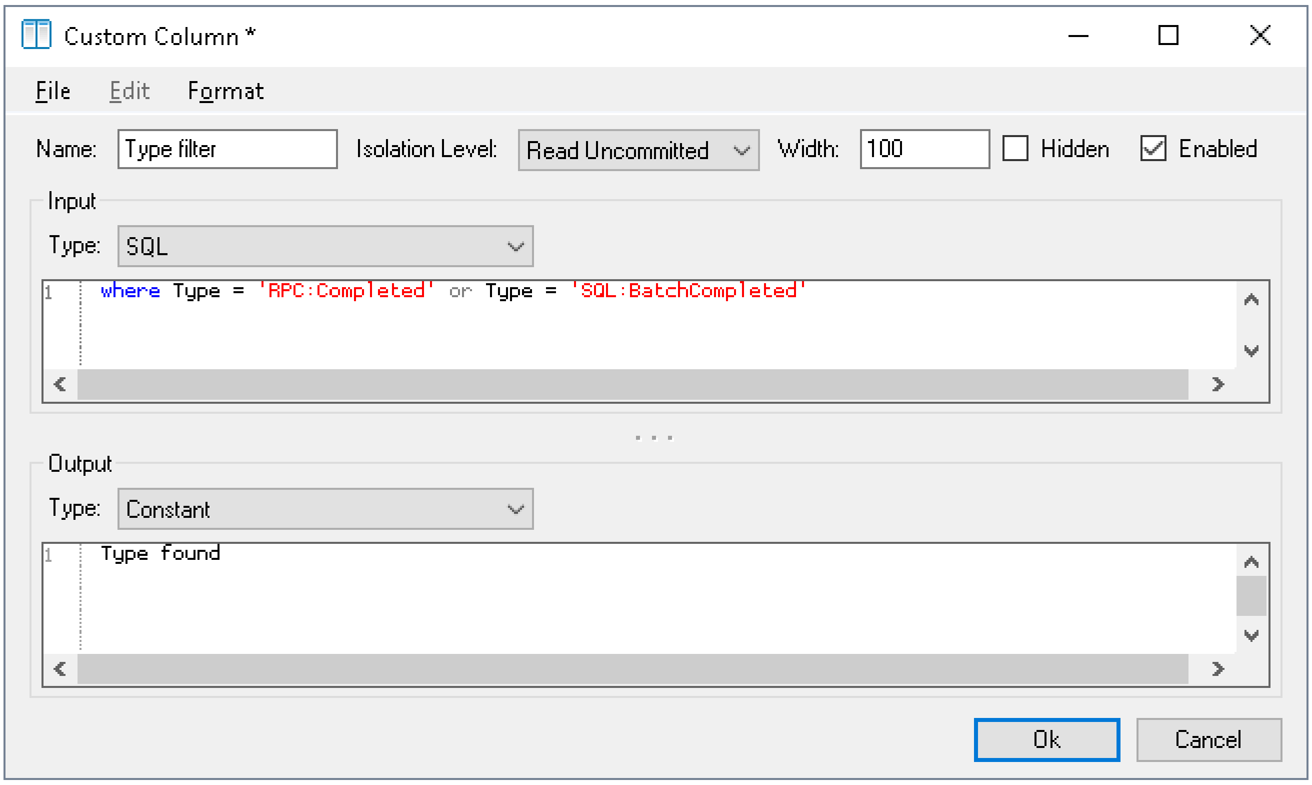
In Filter 1 it is possible to filter on "Type", but it is only possible to choose one value at a time:



Reads between	0	and	72308	☐ Enabled
Writes between	0	and	36	☐ Enabled
CPU between	0	and	16000	☐ Enabled
Rows between	-1	and	10	☐ Enabled
SPID between	52	and	75	☐ Enabled
FileName	All			☐ Enabled
Type	All			☑ Enabled

All
RPC:Completed
SQL:BatchCompleted

Filter 2 can not filter on standard columns, so the task must be solved by creating a custom column:



Name: Type filter
Isolation Level: Read Uncommitted
Width: 100
Hidden [ ]
Enabled [✓]

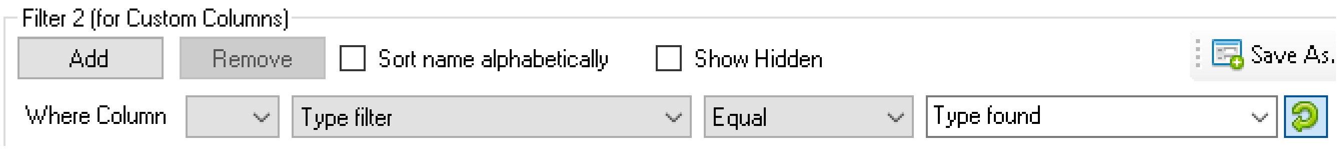
Input Type: SQL

```
where Type = 'RPC:Completed' or Type = 'SQL:BatchCompleted'
```

Output Type: Constant

```
Type found
```

In Filter 2 it is now possible to choose the value from the custom column, and the filtering can be performed:



Where Column | Type filter | Equal | Type found

9. Unattended execution

SQL Event Analyzer can be executed unattended. The unattended mode can be activated by starting SQL Event Analyzer with the following command line parameters:

```
SQLEventAnalyzer.exe [
                     [
                      (-p:"Post Script File")
                      (-e:"Post Script")
                      (-z:"Number of Files")
                      (-d)
                      (-g:"Statistics names" (-f1:"Filter 1 names") (-f2:"Filter
2 names") [(-sp:"Path") or (-sw:"Statistics names")])
                      [
                       [
                        [
                         [-n:Number of Trace Files] or [-f:"Trace File name"]
                        ]
                        (-s:"Session Id" (-x))
                       ]
```

```
                               or

                               [-i:"Session Id" (-x)]
                              ]
                              [-l:"Import Path"]
                             ]

                             or

                             [-u:"Session Id" (-v (-a:"Application name")) (-x)]

                             or

                             [-r]
                            ]
                            (-c:"Custom Column Set File name")
                            (-t:"SQL Server Connection String" (-ms:"SQL"))
                            (-o)
                            (-w:"Web service")
                            (-b:"Database name")
                            (-m:"Text")
```

[]: Mandatory parameter
(): Optional parameter

Parameters:

`(-p:"Post Script File"):`
Full path to SQL Script File.
The script will be executed after importing the Trace Files.

`(-e:"Post Script"):`
SQL Script.
The script will be executed after importing the Trace Files.

`(-z:"Number of Files"):`
Compress Trace Files after import. The newest "Number of Files" will be kept and the rest will be deleted.
Use 0 to keep all files.

`(-d):`
Delete Trace Files after import.

`[-n:Number of Trace Files]:`
The top N most recent Trace Files to import.
Use 0 to import all Trace Files.

`[-f:"Trace File name"]:`
Import all Trace Files newer than the given Trace File.
The events in the given Trace File will not be imported.

`(-s:"Session Id"):`
Append events to specified Session.

If -s is specified, the events will be appended to the specified Session.
If -s is not specified, the events will be imported to a new Session.

`[-i:"Session Id"]:`

Import all Trace Files newer than the last imported Trace File found in
the "FileName" column in the TraceData table for the Session Id specified.

If -i is specified, the events will be appended to the specified Session.

`[-l:"Import Path"]:`

Location of Trace Files to import.

`[-u:"Session Id"]:`

Use the given Session upon start.

`(-v):`

Disable TraceData table changes.

`(-a:"Application name"):`

Use given application name instead of "SQL Event Analyzer".

`(-b:"Database name"):`

Use given application database name instead of "SQLEventAnalyzer".

`(-m:"Text"):`

Add the given text to the execution log.

`(-ms:"SQL"):`

Add output from the given SQL to the execution log. Output will be returned from the first row in the column "Message".
The -t parameter determines the SQL Server connection used.

`(-x):`

Force deletion of the TraceData table for the Session Id specified when exiting.

`[-r]:`

Start in recording mode.

`(-c:"Custom Column Set File name"):`

Full path to file that should be used as Custom Column Set.

`(-t:"SQL Server Connection String"):`

Connect using the given SQL Server Connection String.
If a password is used in the given SQL Server Connection String, it must be encrypted.
If -t is not used, the last successful connection will be used.

`(-o):`

Execution log will be saved in execution path.

`(-w:"Web service"):`

Send execution log to the specified web service.

`(-g:"Statistics names"):`

Generate statistics using the saved statistics with the given names.

Names can be separated by comma.

`(-f1:"Filter 1 names"):`
Apply the Filter 1 with the given names to be used when generating statistics.
Names can be separated by comma.

`(-f2:"Filter 2 names"):`
Apply the Filter 2 with the given names to be used when generating statistics.
Names can be separated by comma.

`(-sw:"Statistics names"):`
Requires the use of -w. Will send the generated statistics with the given names to the given web service.
The statistics names must be included in the names from the -g parameter.

`(-sp:"Path")`
Will save the generated statistics in the given path.

Example 1 (import the two most recent Trace Files and run script afterwards):
```
SQLEventAnalyzer.exe -p:"C:\SomeDir\PostScript.sql" -n:2 -l:"C:\TraceFiles"
```

Example 2 (import all Trace Files newer than "LastImported.trc"):
```
SQLEventAnalyzer.exe -f:"C:\SomeDir\LastImported.trc" -l:"C:\TraceFiles"
```

Example 3 (import all Trace Files newer than the last imported Trace File found in
the "FileName" column in the TraceData table and append the events to an existing Session):
```
SQLEventAnalyzer.exe -i:"collect" -l:"C:\TraceFiles"
```
The post script file and the post script are optional and can be either empty or contain a SQL query that will be executed after the TraceData table has been created and the custom columns has been generated. The SQL query will be executed in the same database where the TraceData table resides.

The post script file and the post script can use parameters by following the same principle as described in the "Parameters" section. E.g. {SessionId} or custom parameters can be used.

It is possible to use the `-p` and `-e` parameters at the same time. In this case, the post script file given with `-p` will be executed first, and the post script given with `-e` will be executed afterwards.

"Trace File name" can be either of the type SQL Server Trace (.trc) or, if supported, Extended Event Trace files (.xel).

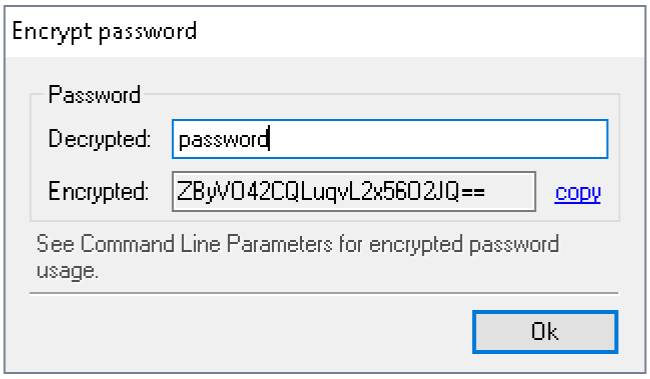
The `-u` parameter can be used to invoke a session directly when starting SQL Event Analyzer. This can be useful if access is only wanted to the events and where trace file import is irrelevant. Furthermore, the `-v` parameter can be given together with `-u` to start SQL Event Analyzer in a mode where it is only possible to see events, but all functionality to modify events are deactivated.

The following functionality are deactivated when using the `-v` parameter:

- The "Custom Columns" menu
- The "Change Connection" menu
- Recording functionality
- The "Back" button
- "Show Hidden Columns"

If the `-z` parameter is used, the imported trace file will be compressed to a 7zip archive.

If the `-t` parameter is used, a password can be used in the given SQL Server Connection String. This password can be encrypted. To encrypt a password, choose "Help", "Command Line Parameters...", "Tools", "Generate encrypted password...":


Encrypt password

Password

Decrypted: password

Encrypted: ZByV042CQLuqvL2x5602JQ==

See Command Line Parameters for encrypted password usage.

If the `-m` parameter is used, the given user defined text will be added to the end of the execution log.

The output will be added to the execution log in the following format:

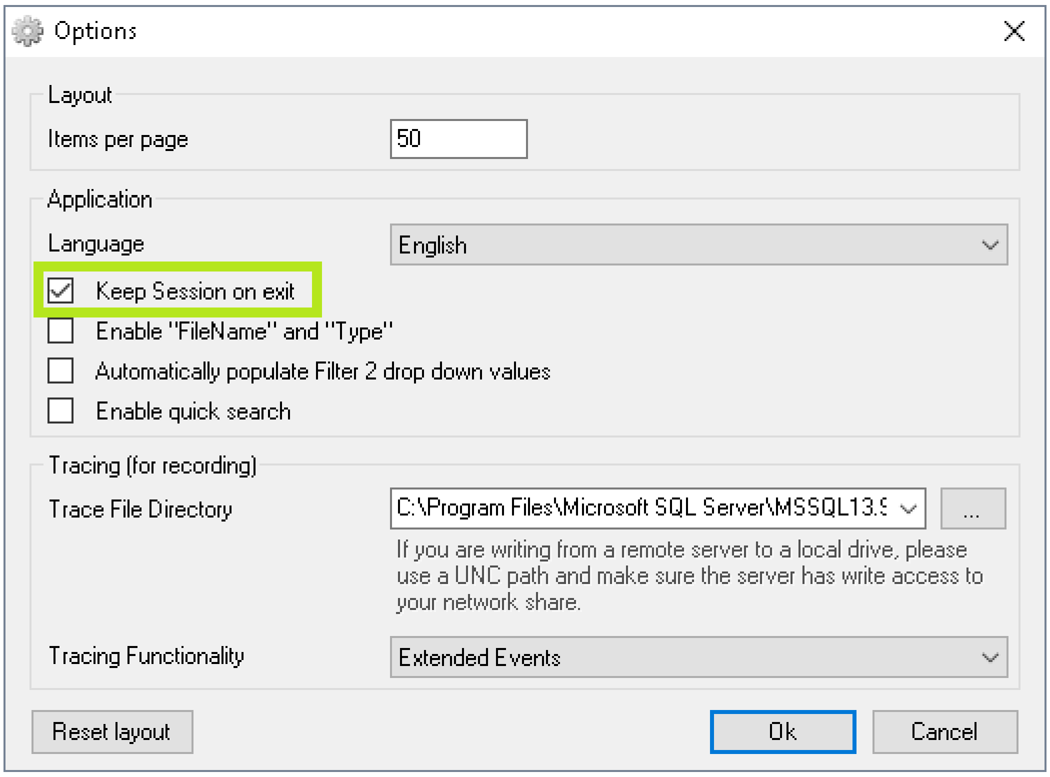
```
2015-05-18T08:48:43: Custom Message:

[BEGIN Custom Message]

...

[END Custom Message]
```

If the `-ms` parameter is used, the output from the given SQL query will be added to the end if the execution log. The name for the output column must be "Message" (without quotes). Output will only be returned from the first row.

The output will be added to the execution log in the following format:

```
2015-05-18T08:48:43: Custom SQL Message:

[BEGIN Custom SQL Message]

...

[END Custom SQL Message]
```

Note, the given session will be deleted if not "Keep Session on exit" is enabled in "Options":


Options

Layout

Items per page: 50

Application

Language: English

[✓] Keep Session on exit

[ ] Enable "FileName" and "Type"

[ ] Automatically populate Filter 2 drop down values

[ ] Enable quick search

Tracing (for recording)

Trace File Directory: C:\Program Files\Microsoft SQL Server\MSSQL13.S

If you are writing from a remote server to a local drive, please use a UNC path and make sure the server has write access to your network share.

Tracing Functionality: Extended Events

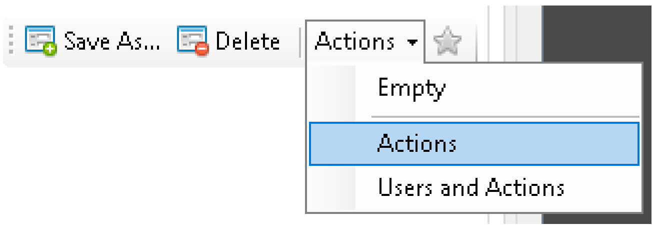
If the `-u`, `-i` or `-s` parameters are used, the session will not be deleted even though "Keep Session on exit" is not enabled. The session can be forced deleted by using the `-x` parameter.

9.1 Automatically generating aggregated statistics

If the `-g` parameter is used together with `-sp` and/or `-sw`, there will be generated aggregated statistics (for more information see the "Statistics" section).

If the `-sw` parameter is used together with the `-g` parameter, the statistics will be send to the web service given in the `-w` parameter (for more information see the "Send execution log to web service" section). If the `-sp` parameter is used, the statistics will be saved in the given path with the file name Statistics.zip.

The statistics used for the automatically generated statistics, is determined from the given name from the `–g` parameter. The name must correspond to a saved search from the Statistics tab:



Here it is possible to create the wanted column groups to be used for the aggregation. To use the above group, "Actions", the `-g` parameter must be given as the following:

```
-g:"Actions"
```

If it is wanted to create multiple statistics for multiple saved searches, these can be given by separating the names of the saved searches with comma:

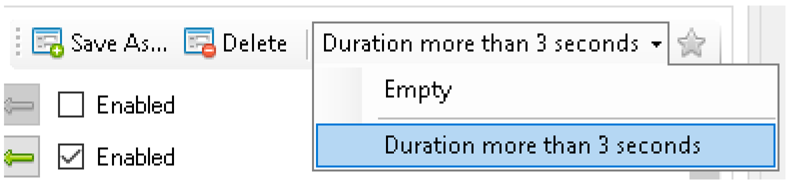
```
-g:"Actions, Users and Actions"
```

It is possible to use Filter 1 and Filter 2 to filter the data used to generating the statistics (for more information see the "Filtered statistics" section).

Filters can be set by using the `-f1` parameter for Filter 1 and the `-f2` parameter for Filter 2. The following shows an example of how to use Filter 1:

```
-f1:"Duration more than 3 seconds"
```

The name of Filter 1 must likewise exist as a saved search for Filter 1:



If multiple statistics are given in the `-g` parameter, optional filters to be used for each statistics can be given likewise by using the `-f1` parameter and the `-f2` parameter. The filter names must be separated by comma. Example:

```
-f2:"Customer Actions, Users in Department ABC"
```

The following example shows three statistics where Filter 1 should be applied on the first statistics and on the last statistics:

```
-g:"SomeStat1, SomeStat2, SomeStat3" -f1:"SomeFilter1,,SomeFilter2"
```

The generated statistics will be saved in the CSV file format, and can be directly opened in Microsoft Excel. The files names are in the following format:

```
[Statistics name][Period begin][Period end][Filter 1 name][Filter 2
name][TotalRows].CSV
```

`[Period begin]` and `[Period end]` defines the begin- and end time for the events that are used as the basis for the statistics.

`[TotalRows]` defines the total number of events that are used as the basis for the statistics.

The final statistics will be delivered in a compressed format in a file with the name Statistics.zip. Statistics.zip contains the wanted statistics in the CSV file format.

The exported statistics will contain all columns, no matter which columns are chosen to be shown in the statistics view. The column order will be determined by the order of the saved statistics searches, and not by the shown column order as when exporting the statistics from the user interface.

The following shows an example of automatically generated statistics of two statistics with the same name ("Actions"), where two different Filter 1 ("All days" and "Last week") are applied, and where the statistics with "Last week" is send to a web service and both statistics are saved to a path:

```
-g:"Actions, Actions" -f1:"All days, Last week" -sw:", Actions" -sp:"C:\somedir"
```

Note the parameter:

```
-sw:", Actions"
```

where it is stated, that it is the second generated statistics (using Filter 1 "Last week") that must be send to the web service.

9.2 Service context execution

If SQL Event Analyzer runs in an unattended mode e.g. from the Windows Task Scheduler, the execution will take place in the service context of the Task Scheduler.

When executing in a service context, the graphical user interface will not be shown. Execution logs from a service context execution can be found in the path where SQL Event Analyzer is installed.

Log files from execution in a service context will be saved from the last 10 executions. When the 11th execution takes place, the oldest of the 10 existing log files will be deleted. The following shows an example of a log file name:

```
SQL Event Analyzer 25082014-110831.log
```

SQL Event Analyzer automatically detects if it is executing in a service context. E.g. if SQL Event Analyzer is started from the command prompt in unattended mode, the graphical user interface will be shown since it is not running in an actual service context.

When executing in a service context, the `–o` parameter will be ignored, since it will always create an execution log in when running in a service context.

9.3 Unattended mode return codes

When executing SQL Event Analyzer in unattended mode, a return code is set when the execution has ended.

Description of the return codes:

Return code	Description
0	Execution successful
-1	Error when running in unattended mode, where SQL Event Analyzer is not started with the -r or -u command line parameters
-2	Error when running in recording mode (command line parameter: -r)
-3	Error when running where a session is set to be used when starting SQL Event Analyzer (command line parameter: -u)
-4	Error when executing post script
-5	Error when deleting imported trace files
-6	Error when importing trace files
-7	Error when compressing imported trace files
-8	Error when communicating with the web service
-9	Error when generating statistics
-10	Error when executing SQL script used for output to the execution log (command line parameter: -ms)

9.4 Send execution log to web service

The execution log can be send to a web service if the `-w` parameter is used.

Signature for the web service communication without an attached file:

```
public bool RegisterLogEvent(string productName, string currentVersion, string machineName,
string userName, string domainName, int statusCode, string message)
```

Signature for the web service communication without an attached file (used if the `-g` and `-sw` parameters are used. For more information see the "Automatically generating aggregated statistics" section):

```
public bool RegisterLogEventWithAttachment(string productName, string currentVersion, string
machineName, string userName, string domainName, int statusCode, string message, byte[]
attachment)
```

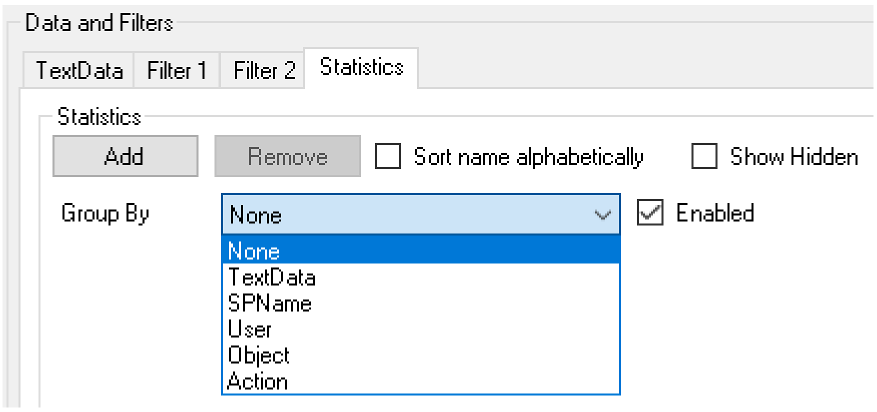
10. Running in recording mode

SQL Event Analyzer can be set to start directly in recording mode. Recording mode can be activated by starting SQL Event Analyzer with the command line parameter `-r` :

```
SQLEventAnalyzer.exe -r
```

11. Statistics

Statistics for the events can be seen from the "Statistics" tab:



Here it is possible to show statistics for user defined grouping of data.

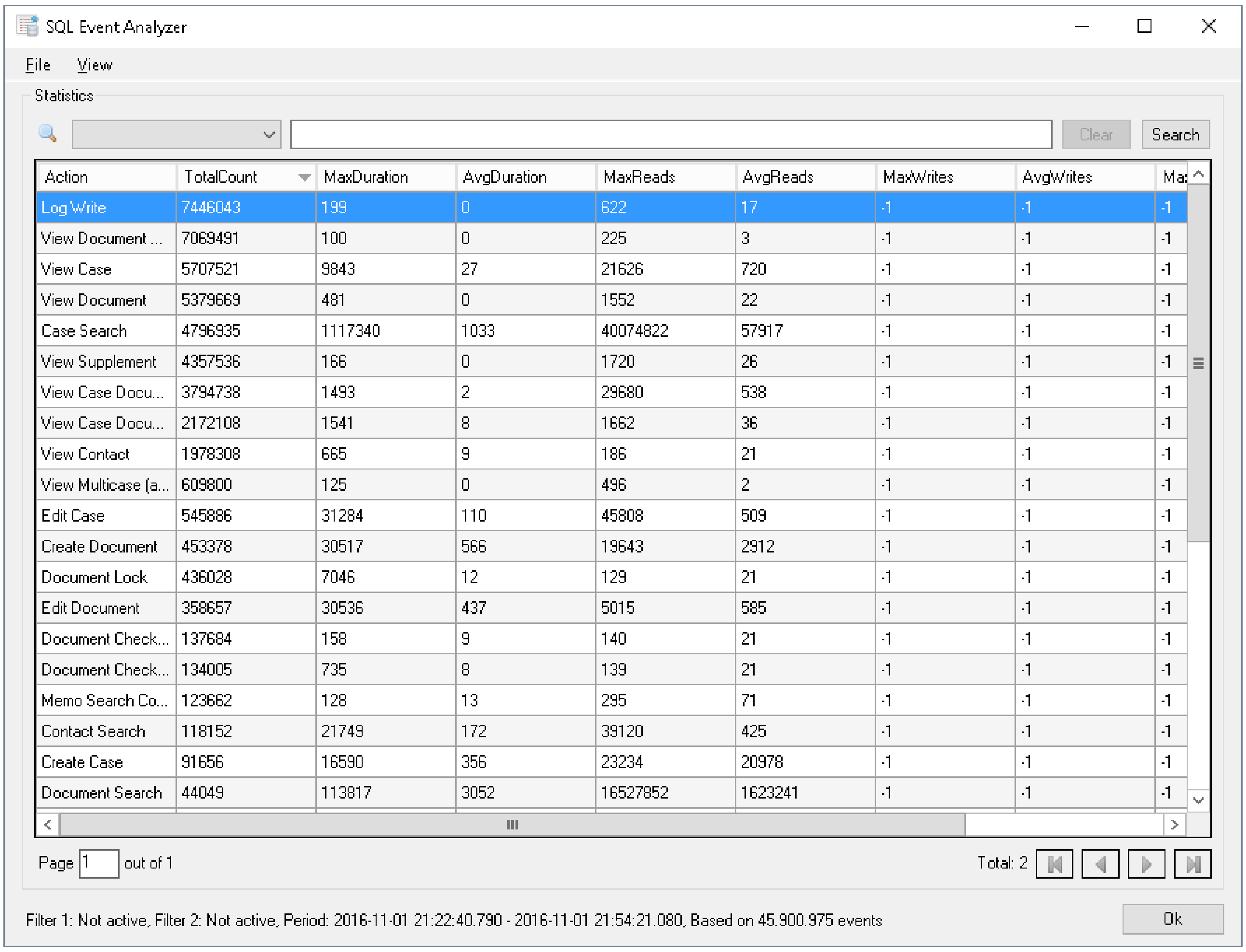
If no grouping is chosen, the overall statistics for all events are shown.

Note, "TextData" will always be at the top of the list and is not influenced by "Sort name alphabetically".

Likewise Filter 1 and Filter 2, searches can be saved, deleted and modified (for more information se the "Filters" section).

Example:

Show statistics grouped by the custom column "Action":



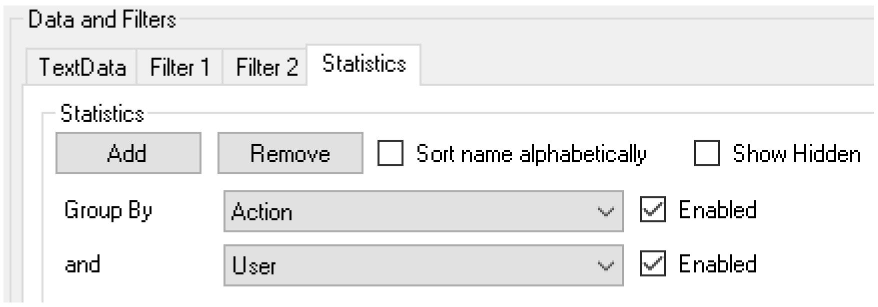
Action	TotalCount	MaxDuration	AvgDuration	MaxReads	AvgReads	MaxWrites	AvgWrites	Ma
Log Write	7446043	199	0	622	17	-1	-1	-1
View Document ...	7069491	100	0	225	3	-1	-1	-1
View Case	5707521	9843	27	21626	720	-1	-1	-1
View Document	5379669	481	0	1552	22	-1	-1	-1
Case Search	4796935	1117340	1033	40074822	57917	-1	-1	-1
View Supplement	4357536	166	0	1720	26	-1	-1	-1
View Case Docu...	3794738	1493	2	29680	538	-1	-1	-1
View Case Docu...	2172108	1541	8	1662	36	-1	-1	-1
View Contact	1978308	665	9	186	21	-1	-1	-1
View Multicase (a...	609800	125	0	496	2	-1	-1	-1
Edit Case	545886	31284	110	45808	509	-1	-1	-1
Create Document	453378	30517	566	19643	2912	-1	-1	-1
Document Lock	436028	7046	12	129	21	-1	-1	-1
Edit Document	358657	30536	437	5015	585	-1	-1	-1
Document Check...	137684	158	9	140	21	-1	-1	-1
Document Check...	134005	735	8	139	21	-1	-1	-1
Memo Search Co...	123662	128	13	295	71	-1	-1	-1
Contact Search	118152	21749	172	39120	425	-1	-1	-1
Create Case	91656	16590	356	23234	20978	-1	-1	-1
Document Search	44049	113817	3052	16527852	1623241	-1	-1	-1

Page 1 out of 1

Total: 2

Filter 1: Not active, Filter 2: Not active, Period: 2016-11-01 21:22:40.790 - 2016-11-01 21:54:21.080, Based on 45.900.975 events

It is possible to group by as many custom columns as wanted:



☐ Sort name alphabetically ☐ Show Hidden

Group By: Action ☑ Enabled

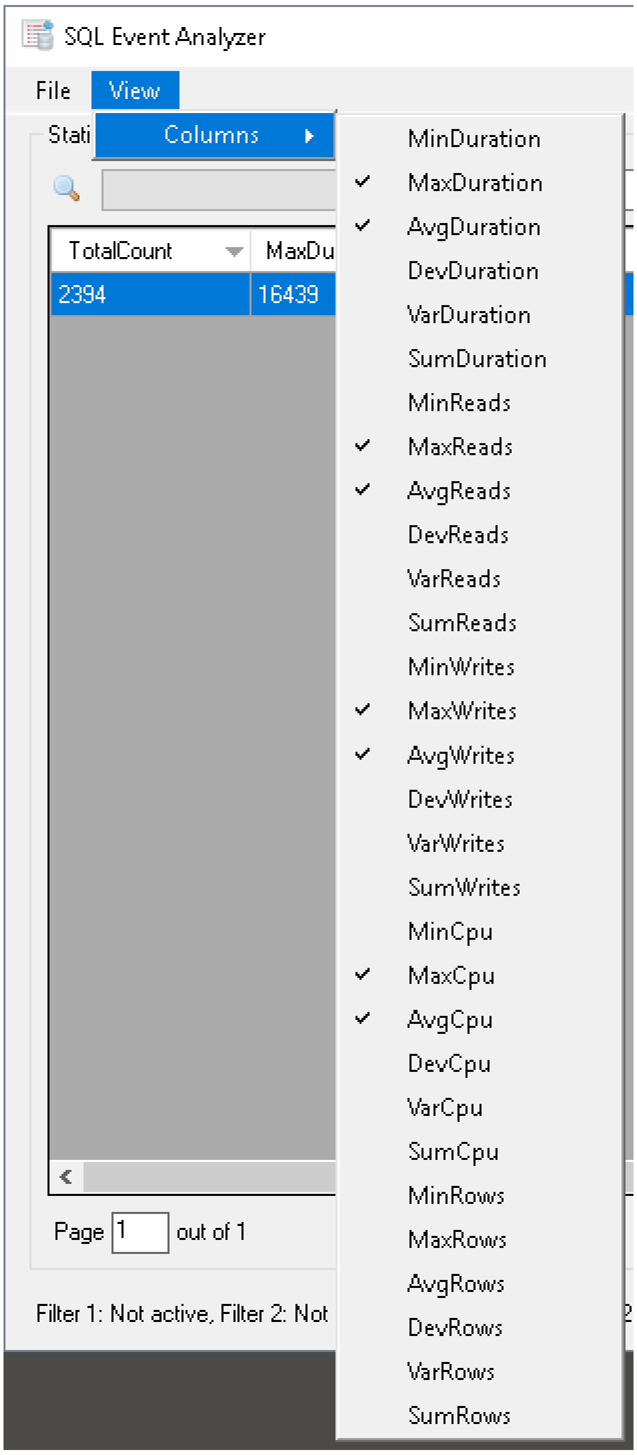
and: User ☑ Enabled

The following statistical data can be shown:

• TotalCount:	Count
• MinDuration:	Minimum execution time
• MaxDuration:	Maximum execution time
• AvgDuration:	Average execution time
• DevDuration:	Standard deviation in execution time
• VarDuration:	Variance in execution time
• SumDuration:	Sum of execution time
• MinReads:	Minimum Reads
• MaxReads:	Maximum Reads
• AvgReads:	Average Reads
• DevReads:	Standard deviation in Reads
• VarReads:	Variance in Reads
• SumReads:	Sum of Reads
• MinWrites:	Minimum Writes
• MaxWrites:	Maximum Writes
• AvgWrites:	Average Writes
• DevWrites:	Standard deviation in Writes
• VarWrites:	Variance in Writes
• SumWrites:	Sum of Writes
• MinCPU:	Minimum CPU
• MaxCPU:	Maximum CPU
• AvgCPU:	Average CPU
• DevCPU:	Standard deviation in CPU
• VarCPU:	Variance in CPU
• SumCPU:	Sum of CPU
• MinRows:	Minimum Rows
• MaxRows:	Maximum Rows
• AvgRows:	Average Rows
• DevRows:	Standard deviation in Rows
• VarRows:	Variance in Rows
• SumRows:	Sum of Rows

Execution times are in milliseconds.

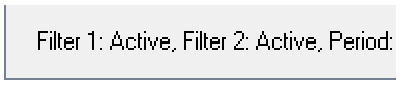
Not all columns are shown by default. Which columns to show can be chosen in "View", "Columns":



11.1 Filtered statistics

Filter 1 and Filter 2 can be used to filter the data used for the statistics.

In the bottom left corner on in the statistics view, is shown which filters are active:



Filter 1: Active, Filter 2: Active, Period:

11.2 Improve performance when working with statistics

If working with large data sets, is it advised to use a Columnstore Index on the TraceData table. This is only possible if using SQL Server 2012 or newer.

A Columnstore Index can manually be created by executing the following:

```
create nonclustered columnstore index ix_CS on dbo.[TraceData_{SessionId}] (ID, [User],
[Object], [Action], Duration, Reads, Writes, CPU, Rows)
```

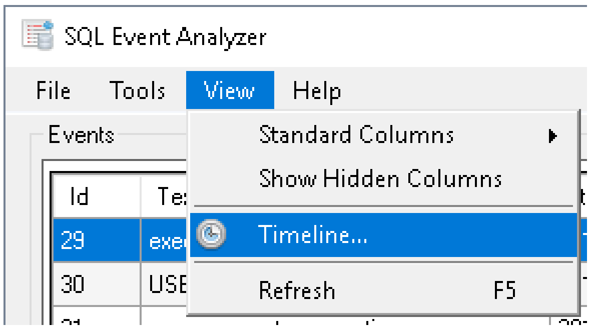
Where `{SessionId}` must be replaced with the correct session id.

In the above script, the custom columns `[User], [Object], [Action]` are used. These can freely be changed.

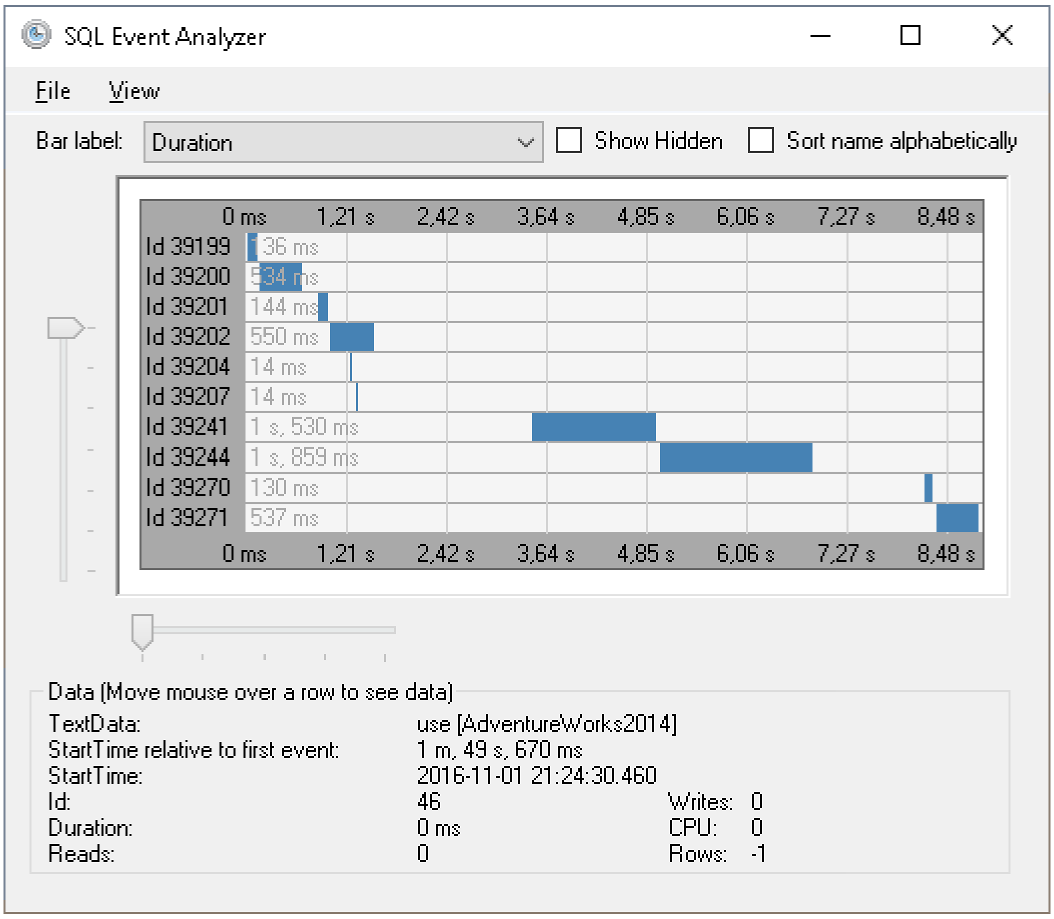
The name for the Columnstore Index must be `ix_CS`. If another name is used, SQL Event Analyzer will fail.

12. Timeline

A timeline for the events can be seen from "View", "Timeline...":



The timeline:

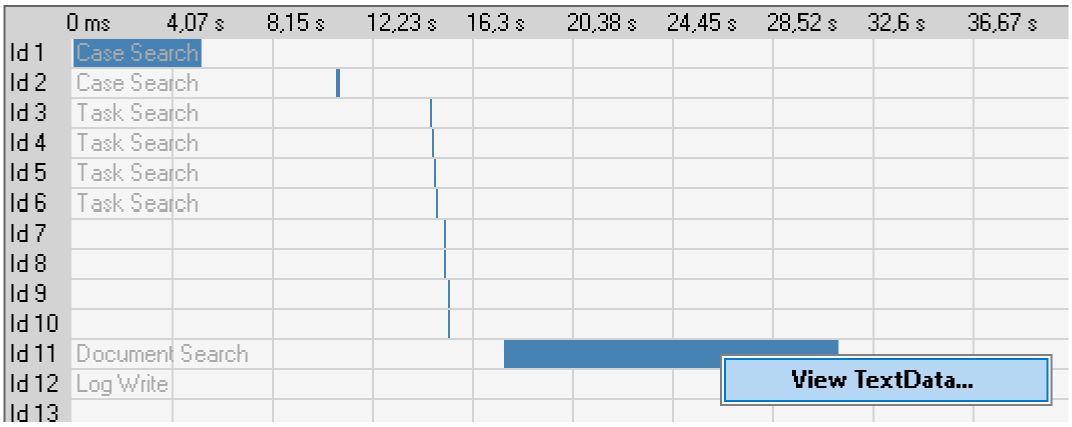


Data (Move mouse over a row to see data)

TextData:	use [AdventureWorks2014]		
StartTime relative to first event:	1 m, 49 s, 670 ms		
StartTime:	2016-11-01 21:24:30.460		
Id:	46	Writes:	0
Duration:	0 ms	CPU:	0
Reads:	0	Rows:	-1

Information about events can be seen by hovering the mouse above the individual events.

It is possible to right click on an event and see the SQL statement for the event:

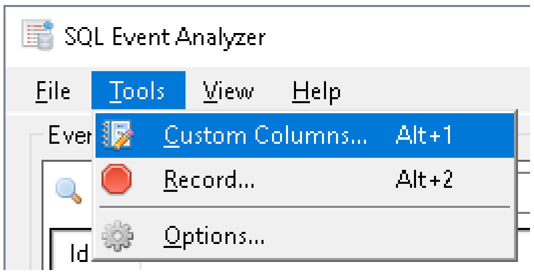


Note, that the timeline can not show more than 1000 events, and only show events where the time span between the first- and last event is maximum 1 day.

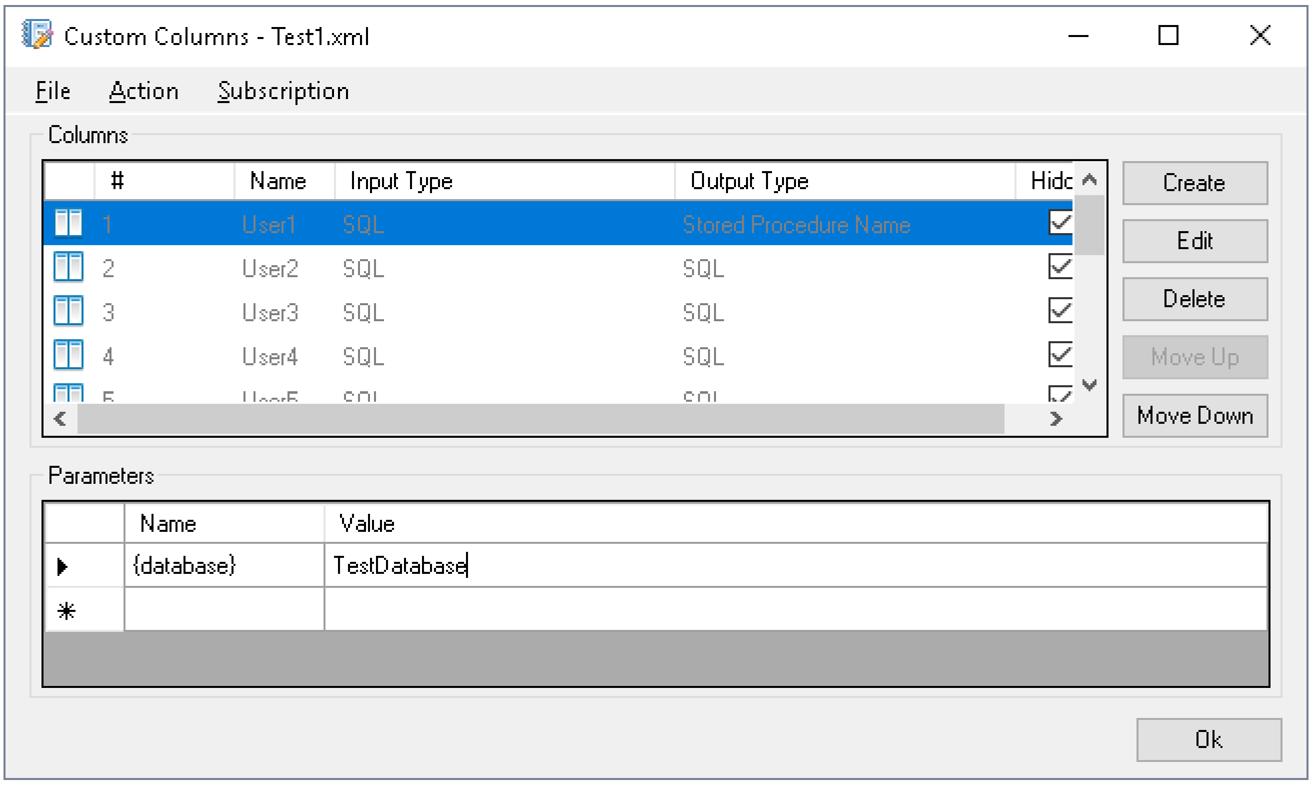
13. Parameters

Parameters can be used to replace values in the text for input- and output types.

Parameters can be set in "Tools", "Custom Columns...":



In the "Parameters" section, parameters can be added, modified and deleted:

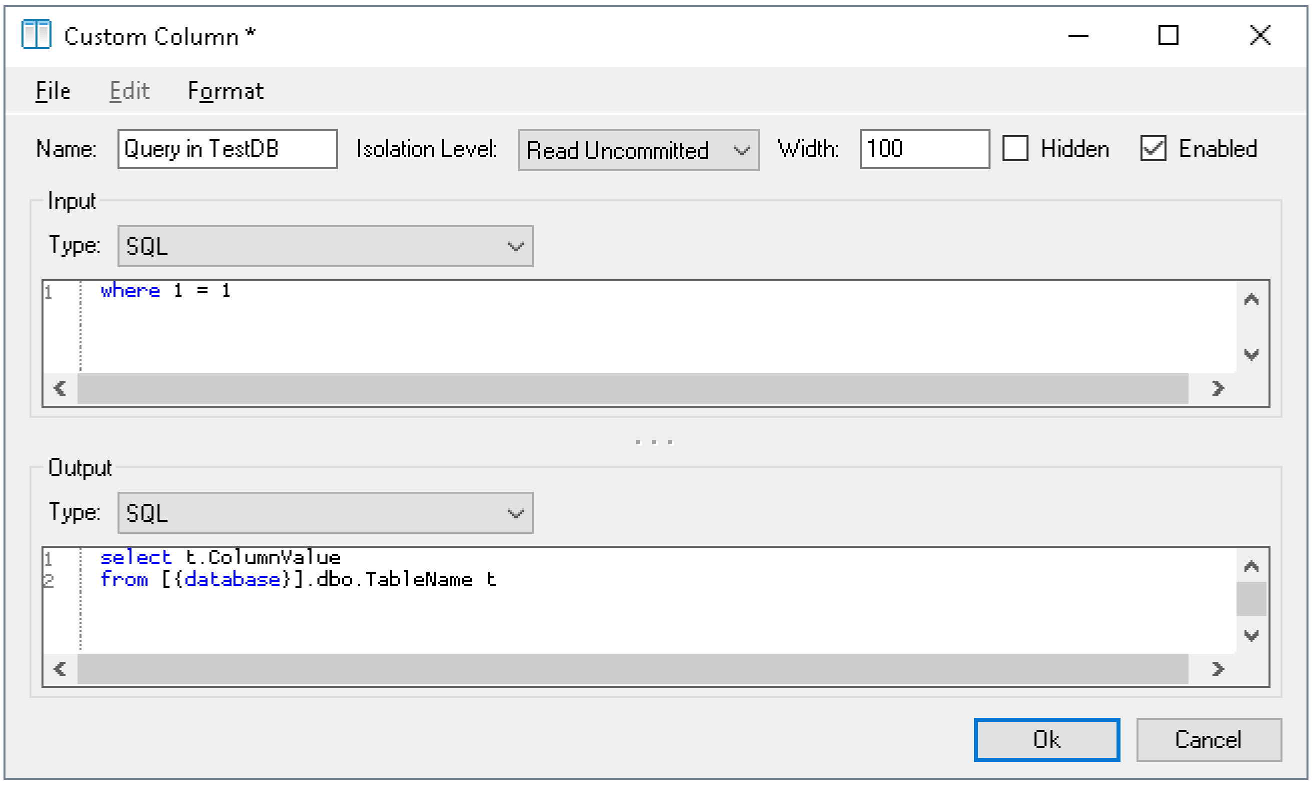


#	Name	Input Type	Output Type	Hidc
1	User1	SQL	Stored Procedure Name	✓
2	User2	SQL	SQL	✓
3	User3	SQL	SQL	✓
4	User4	SQL	SQL	✓

	Name	Value
▶	{database}	TestDatabase
*		

The values given in "Name" will be replaced with the values in "Value" when handling the columns. In the above example, "{database}" will be replaced with "TestDatabase".

The following shows an example of the use of a parameter value in the output text:



Name: Query in TestDB
Isolation Level: Read Uncommitted
Width: 100
Hidden [✓ Enabled]

Input
Type: SQL

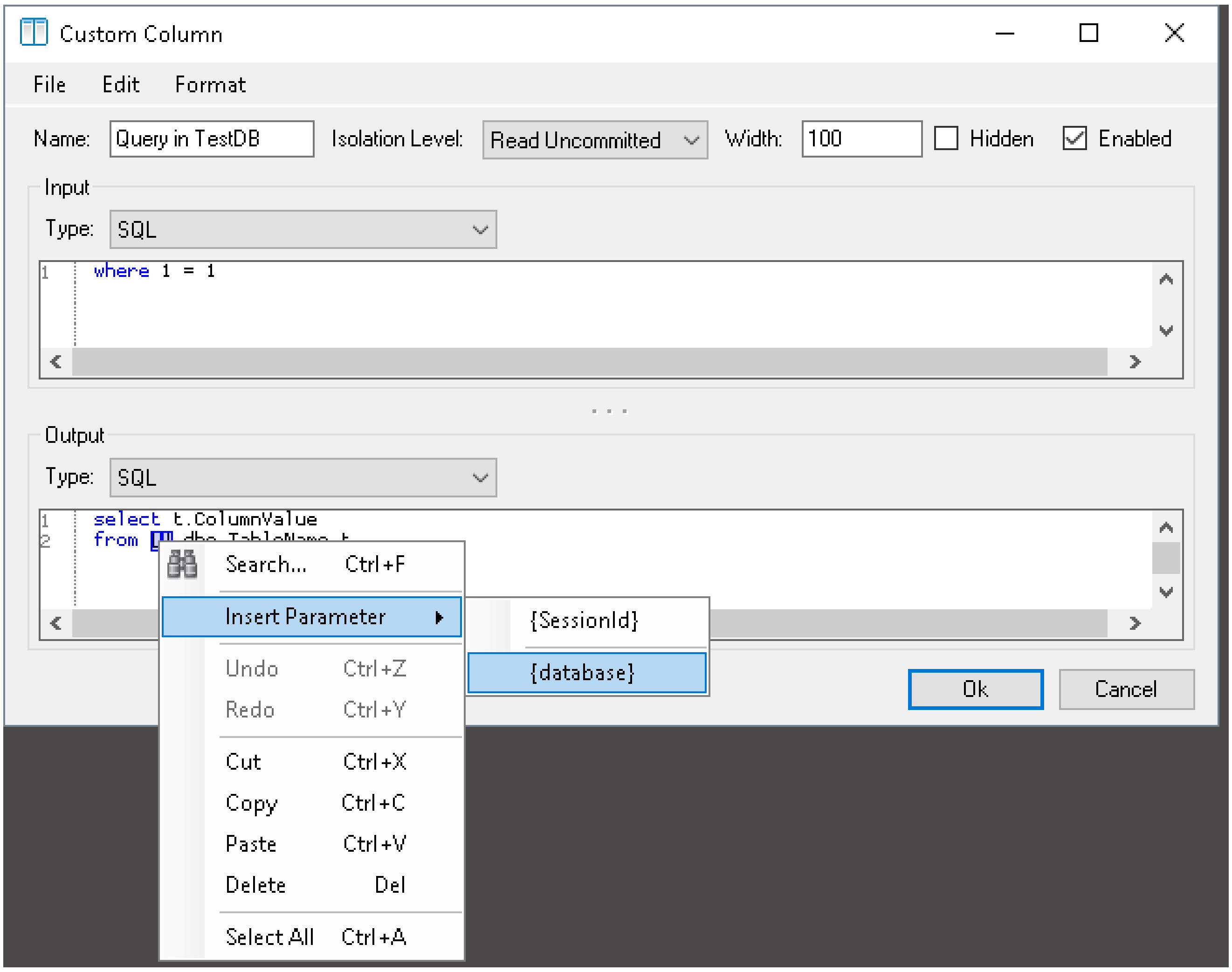
```
where 1 = 1
```

Output
Type: SQL

```
select t.ColumnValue
from [{database}].dbo.TableName t
```

Note, the values for "Name" and "Value" can freely be chosen and do not have to be encapsulated by { }.

Given parameters can be insert directly from the right click menu:



```
where 1 = 1
```

```
select t.ColumnValue
from [] dbo.TableName t
```

The {SessionId} parameter is a standard parameter which contains the active session id.