

How to Run SQL Profiler Trace to Capture Information (Internal)

07/16/2025 6:00 pm EDT

Issue:

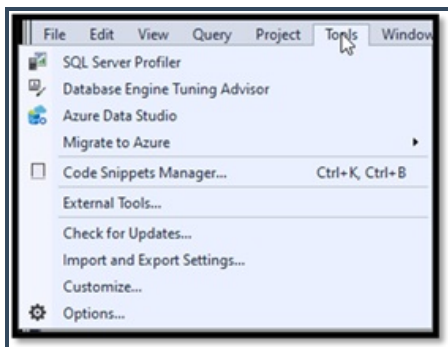
The customer is receiving errors in their SedonaOffice company. Is there a way to get more information on why the errors are occurring?

Resolution:

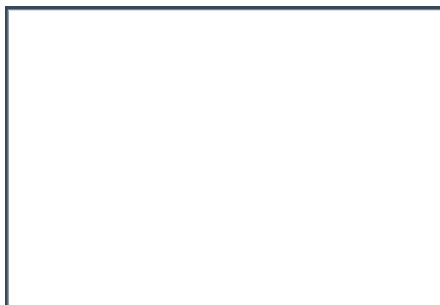
SQL Server Profiler can be used to capture information from processes accessing the SQL Server databases. Using SQL Server Profiler, you can create a trace that records and displays the commands sent to and the responses from the SQL server.

Open SQL Server Management Studio.

Connect to the correct server, select Tools, then SQL Server Profiler.

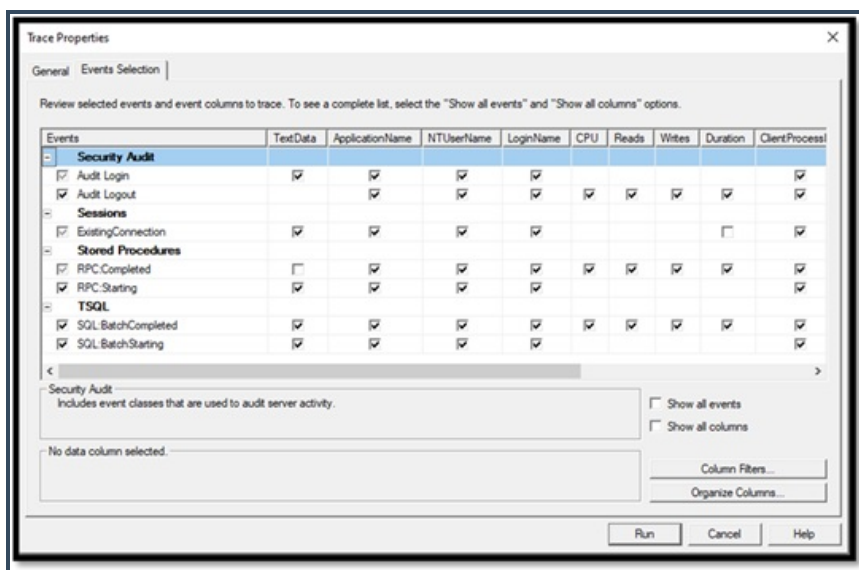


It can also be launched from the Windows Menu.



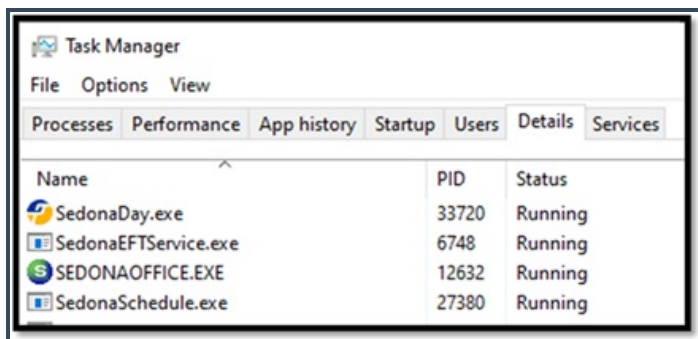
In the Events Selection Tab, you'll see the events it will monitor.

Make sure the options selected are the same as in the screenshot below.

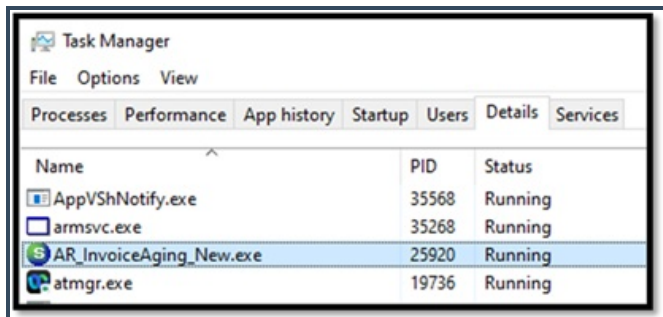


To limit the results to the machine having the issue, we use the ProcessID from the application. This can be obtained from the Windows Task Manager on the Details tab.

If the problem is with a report or Service Ticket, you will need to look for the .exe for the particular report or the SedonaScheduler.



If the problem is a report or any other application that opens outside of the SedonaOffice window, look for a separate application in the Task Manager. You may need to use this instead of SedonaOffice.exe.



Enter the PID from the application in the Column Filter for ClientProcessID.



Click OK.

Once the events and filters are correct, click Run on the Trace Properties.

Start the trace right before performing the action.

Once running, it will capture the activity between the application and the server.

Perform the action in the application that is causing the error. Once the error is received, stop the Trace and save the file.

The trace will show every transaction between the application and server and can be used to see which event caused the errors. The saved trace file can be attached to the support case or emailed to Support for analysis.

EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Duration
RPC:Starting	exec JobQueue_Get '','','','0','...	SedonaOffice...		Sedona...				
RPC:Completed	exec JobQueue_Get '','','','0','...	SedonaOffice...		Sedona...	32	5817	4	62
RPC:Starting	declare @p4 money set @p4=NULL exe...	SedonaOffice...		Sedona...				
RPC:Completed	declare @p4 money set @p4=50.0000 ...	SedonaOffice...		Sedona...	0	9	0	0
RPC:Starting	exec Critical_Messages_Get 8,1	SedonaOffice...		Sedona...				
RPC:Completed	exec Critical_Messages_Get 8,1	SedonaOffice...		Sedona...	0	212	0	1
RPC:Starting	declare @p1 int set @p1=NULL decla...	SedonaOffice...		Sedona...				
RPC:Completed	declare @p1 int set @p1=1 declare ...	SedonaOffice...		Sedona...	0	51	0	1
RPC:Starting	exec sp_unprepare 1	SedonaOffice...		Sedona...				
RPC:Completed	exec sp_unprepare 1	SedonaOffice...		Sedona...	0	0	0	0
RPC:Starting	declare @p1 int set @p1=NULL decla...	SedonaOffice...		Sedona...				
RPC:Completed	declare @p1 int set @p1=1 declare ...	SedonaOffice...		Sedona...	0	9	0	0

Trace is running.

Ln 799, Col 2 Rows: 804

If the trace is stopped as soon as the error occurs, you can see the last transactions that caused the error.

You can select the line in the trace to see what transaction took place.

SQLBatchCompleted	SELECT ac.Alarm_Company_Code, ac.Des...	SedonaOffice...	Sedona...	0	28	0
SQLBatchStarting	SELECT site.Customer_Site_Id, site.A...	SedonaOffice...	Sedona...			
SQLBatchCompleted	SELECT site.Customer_Site_Id, site.A...	SedonaOffice...	Sedona...	0	53	0
SQLBatchStarting	SELECT cust.Customer_System_ID, sys...	SedonaOffice...	Sedona...			
SQLBatchCompleted	SELECT cust.Customer_System_ID, sys...	SedonaOffice...	Sedona...	15	169	0


```

SELECT ac.Alarm_Company_Code, ac.Description, ac.Inactive, isnull(ac.Integration_Id, 1) as Integration_Id, ac.Alarm_Company_Id FROM
CS_Alarm_Company ac WHERE ac.Inactive = 'N' order by ac.Alarm_Company_Code

```

You can copy and paste this in an SSMS Query window to execute the transaction to see if there is a problem.

SQLQuery13.sql - BG - cortt.Pickens (59) | NO SQL Windowing - cortt.Pickens (73) | List triggers and ts - Scott.Pickens (59) | SQLQuery13.sql - BG - cortt.Pickens

```

SELECT ac.Alarm_Company_Code, ac.Description, ac.Inactive, isnull(ac.Integration_Id, 1) as Integration_Id, ac.Alarm_Company_Id
FROM CS_Alarm_Company ac WHERE ac.Inactive = 'N' order by ac.Alarm_Company_Code

```

100 % | Results | Messages

	Alarm_Company_Code	Description	Inactive	Integration_Id	Alarm_Company_Id
1	Alamnet	Alamnet	N	1	3
2	CMS	CMS	N	4	7
3	Manitou	Manitou	N	8	10
4	Manitou	Manitou	N	1	4
5	N/A	N/A	N	1	1
6	RapidResponse	RapidResponse	N	1	6
7	SedonaSecurity	SedonaSecurity	N	1	2
8	Stages	IntStages	N	5	8
9	XYZ Monitoring	XYZ Monitoring	N	1	5

If it is a stored procedure, you can also copy and paste the statement. Be careful of procedures that are an Update, Add, Delete, or an Insert. If executed, these may add or change records that should not be changed.

Untitled - 1 (BG-DSCZ793)

EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Dur
SQLBatchCompleted	DELETE FROM AR_Customer_Bill_Email W...	SedonaOffice...		Sedona...	0	36	1	
RPC:Completed	declare @pi int set @pi=5 declare ...	SedonaOffice...		Sedona...	0	33	1	

```

declare @pi int
set @pi=5
declare @pi0 int
set @pi0=177
exec sp_prepekecrpc @pi output, N'sp_executesql', N'INSERT INTO AR_Customer_Bill_Email
(Customer_Bill_Id, Email_Address,
is_Primary, Invalid, InActive)
VALUES(@billId, @EmailAddress, @isPrimary, @Invalid, @Inactive);SET @EmailId =
SCOPE_IDENTITY();', N'@billId int, @EmailAddress nvarchar(50), @isPrimary char(1), @Invalid char(1), @Inactive char(1), @EmailId int
OUTPUT', $3976, N'scott.pickens@boldgroup.com', 'Y', 'N', 'N', @pi0 output
select @pi, @pi0

```

If you need to test something like this, always use the Sandbox company.

If it is a Select or a stored procedure to GET information, it is OK to run in the production company. This is OK to run since it is a stored procedure to get the Master Account information.

Untitled - 1 (BG-DSCZ793)

EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Dur
SQLBatchCompleted	SELECT cust.Customer_System_ID, sys...	SedonaOffice...		Sedona...	0	152	0	
SQLBatchStarting	SELECT cust.Customer_System_ID, sys...	SedonaOffice...		Sedona...				
SQLBatchCompleted	SELECT cust.Customer_System_ID, sys...	SedonaOffice...		Sedona...	16	596	0	
SQLBatchStarting	SELECT cust.Customer_System_ID, sys...	SedonaOffice...		Sedona...				
SQLBatchCompleted	SELECT cust.Customer_System_ID, sys...	SedonaOffice...		Sedona...	16	152	0	
SQLBatchStarting	SELECT site.Customer_Site_Id, site.A...	SedonaOffice...		Sedona...				
SQLBatchCompleted	SELECT site.Customer_Site_Id, site.A...	SedonaOffice...		Sedona...	0	4	0	
SQLBatchStarting	SELECT b.Customer_Id, c.Customer_Num...	SedonaOffice...		Sedona...				
SQLBatchCompleted	SELECT b.Customer_Id, c.Customer_Num...	SedonaOffice...		Sedona...	0	31	0	
RPC:Completed	declare @p4 int set @p4=1 declare ...	SedonaOffice...		Sedona...	0	10	0	

```

declare @p4 int
set @p4=1
declare @p5 varchar(25)
set @p5='N/A'
declare @p6 varchar(50)
set @p6='N/A'
declare @p7 varchar(15)
set @p7='00000'
declare @p8 char(1)
set @p8='N'
declare @p9 datetime
set @p9='1899-12-30 00:00:00'
declare @p10 int
set @p10=81
declare @p11 char(1)
set @p11='Y'
exec Customer_Master_Account_GET '1079', 1, 2, @p4 output, @p5 output, @p6 output, @p7 output, @p8 output, @p9 output, @p10 output, @p11 output
select @p4, @p5, @p6, @p7, @p8, @p9, @p10, @p11

```

This will help narrow down the location of the problem.

