How to get some audit information without auditing

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:33 AM PST

Contents

Important note

The method below can be used but be aware that is not officially supported - the DMVs that are presented below are officially only supported on SQL Server. Also this DMVs will be retired in the future

Use this method as a **best effort** to help customers in case of not having auditing enabled. Suggesting auditing and/or Xevents should always be recommended.

Note that the method below is not guaranteed to return data for various reasons:

- the trace file rolled over
- the managed instance failover and now is on a new node with a different default trace
- the DMVs simply were retired

What is the default trace

The default trace is a log enabled by the default that records various activities that can be used for auditing.

It is saved on the SQL Server

Note that the default trace will be removed in the future. The usage of the DMVs below is only for helping customers on a **best effort** basis.

Customers should enable auditing or xtended events.

DMVs used and information returned

To read the default trace we will need a few DMVs:

sys.traces - returns running traces. We will use it to get the running traces names.

sys.trace_events - list of trace events

sys.fn_trace_gettable - returns a trace file on tabular form.

Example

First of all, let's see the events that are available on a trace:

```
SELECT DISTINCT
 e.trace_event_id ,
 e.name
FROM sys.fn_trace_geteventinfo (1) t
 JOIN sys.trace_events e
 ON t.eventID = e.trace_event_id
```

```
trace_event_id name
-------------- ----------------------------------------------------------------------------------------------------
18             Audit Server Starts And Stops
20             Audit Login Failed
22             ErrorLog
46             Object:Created
47             Object:Deleted
55             Hash Warning
69             Sort Warnings
79             Missing Column Statistics
80             Missing Join Predicate
81             Server Memory Change
92             Data File Auto Grow
93             Log File Auto Grow
94             Data File Auto Shrink
95             Log File Auto Shrink
102            Audit Database Scope GDR Event
103            Audit Schema Object GDR Event
104            Audit Addlogin Event
105            Audit Login GDR Event
106            Audit Login Change Property Event
108            Audit Add Login to Server Role Event
109            Audit Add DB User Event
110            Audit Add Member to DB Role Event
111            Audit Add Role Event
115            Audit Backup/Restore Event
116            Audit DBCC Event
117            Audit Change Audit Event
152            Audit Change Database Owner
153            Audit Schema Object Take Ownership Event
155            FT:Crawl Started
156            FT:Crawl Stopped
164            Object:Altered
167            Database Mirroring State Change
175            Audit Server Alter Trace Event
218            Plan Guide Unsuccessful
```

Some of this entries might be helpful. For example, a table that was dropped. There is no auditing and I want to know who dropped the table.

First, let's create and drop a table for testing purposes.

```
create table testingtable (id int)
go
drop table testingtable
go
```

Now let's find out who deleted the table. Looking at the events available, I will want event ID 47, since I want to look at object deletion events:

```
Declare @physical_db_name nvarchar(255)
select @physical_db_name = physical_database_name from sys.databases
where name = 'archetype_rimarqu'; -- change the database name

DECLARE @FileName NVARCHAR(260)
SELECT @FileName = SUBSTRING(path, 0,
 LEN(path) - CHARINDEX('\',
 REVERSE(path)) + 1)
 + '\Log.trc'
FROM sys.traces
WHERE is_default = 1 ;
SELECT loginname ,
 hostname ,
 applicationname ,
 databasename ,
 objectName ,
 starttime ,
 e.name AS EventName ,
 databaseid
FROM sys.fn_trace_gettable(@FileName, DEFAULT) AS gt
 INNER JOIN sys.trace_events e
 ON gt.EventClass = e.trace_event_id
 WHERE (gt.EventClass = 47) -- event id for object creation
and databasename = @physical_db_name
 order by StartTime desc
```

This returns the hostname, login, name of the object deleted and program used.

Results | Messages

	loginname	hostname	applicationname	databasename	objectName	starttime	EventName	databaseid
1	rimarqu	[redacted]	Microsoft SQL Server Management Studio - Query	50c7b16d-9e2b-48d6-a8f4-401b16a469cd	testingtable	2023-01-10 11:35:23.267	Object:Deleted	16
2	rimarqu	[redacted]	Microsoft SQL Server Management Studio - Query	50c7b16d-9e2b-48d6-a8f4-401b16a469cd	testingtable	2023-01-10 11:35:23.263	Object:Deleted	16
3	miadmin	[redacted]	Microsoft SQL Server Management Studio - Query	50c7b16d-9e2b-48d6-a8f4-401b16a469cd	testing	2023-01-10 11:31:08.590	Object:Deleted	16

Now, let's imagine the following scenario:

1 - one user created a table and inserted records

```
create table testingtableidentity (id int identity(1,1), col1 varchar(50))
go
insert into testingtableidentity values ('qwerty')
go 100
```

2 - another user created a new column and populated the new column. Also reseeded the identity.

```
alter table testingtableidentity
add col2 varchar(100)
go
update testingtableidentity set col2 = 'bbbbbbb'
go
DBCC CHECKIDENT ('testingtableidentity', RESEED, 0);
```

We want to get more details on the actions from point 2.

Just for learning purposes, let's use the object name instead of Event id's:

```
Declare @physical_db_name nvarchar(255)
select @physical_db_name = physical_database_name from sys.databases
where name = 'archetype_rimarqu'; -- change the database name

DECLARE @FileName NVARCHAR(260)
SELECT @FileName = SUBSTRING(path, 0,
 LEN(path) - CHARINDEX('\',
 REVERSE(path)) + 1)
 + '\Log.trc'
FROM sys.traces
WHERE is_default = 1 ;
SELECT loginname ,
 hostname ,
 applicationname ,
 databasename ,
 objectName ,
 starttime ,
 e.name AS EventName ,
 databaseid
FROM sys.fn_trace_gettable(@FileName, DEFAULT) AS gt
 INNER JOIN sys.trace_events e
 ON gt.EventClass = e.trace_event_id
-- WHERE (gt.EventClass = 116) -- event id
where ObjectName = 'testingtableidentity'
and databasename = @physical_db_name
 order by StartTime desc
```

So we have the two events - one user created the table and another altered the table

100 %

Results | Messages

	loginname	hostname	applicationname	databasename	objectName	starttime	EventName	databaseid
1	rimarqu	[redacted]	Microsoft SQL Server Management Studio - Query	50c7b16d-9e2b-48d6-a8f4-401b16a469cd	testingtableidentity	2023-01-10 11:44:57.973	Object:Altered	16
2	rimarqu	[redacted]	Microsoft SQL Server Management Studio - Query	50c7b16d-9e2b-48d6-a8f4-401b16a469cd	testingtableidentity	2023-01-10 11:44:57.967	Object:Altered	16
3	miadmin	[redacted]	Microsoft SQL Server Management Studio - Query	50c7b16d-9e2b-48d6-a8f4-401b16a469cd	testingtableidentity	2023-01-10 11:44:07.477	Object:Created	16
4	miadmin	[redacted]	Microsoft SQL Server Management Studio - Query	50c7b16d-9e2b-48d6-a8f4-401b16a469cd	testingtableidentity	2023-01-10 11:44:07.470	Object:Created	16

Note that DML operations are not tracked on the default trace, like so is not expected to see the first Insert and the Update made by the second user.

Now, we are missing the reseed of the identity, since it`s not a change of an object but a DBCC command.

Let's search by event 116 (Audit DBCC Audit):

```
Declare @db_id int
select @db_id = database_id from sys.databases
where name = 'archetype_rimarqu'; -- change the database name

DECLARE @FileName NVARCHAR(260)
SELECT @FileName = SUBSTRING(path, 0,
 LEN(path) - CHARINDEX('\',
 REVERSE(path)) + 1)
 + '\Log.trc'
FROM sys.traces
WHERE is_default = 1 ;
SELECT loginname ,
 hostname ,
 applicationname ,
 databasename ,
 objectName ,
 starttime ,
 e.name AS EventName ,
 databaseid, TextData
FROM sys.fn_trace_gettable(@FileName, DEFAULT) AS gt
 INNER JOIN sys.trace_events e
 ON gt.EventClass = e.trace_event_id
 WHERE (gt.EventClass = 116) -- event id
--where ObjectName = 'testingtableidentity'
and databaseid = @db_id
 order by StartTime desc
```

This returns the identity reseed:

Results Messages

	loginname	hostname	applicationname	databasename	objectName	starttime	EventName	databaseid	TextData
1	rimarqu	[redacted]	Microsoft SQL Server Management Studio - Query	archetype_rimarqu	NULL	2023-01-10 11:49:14.310	Audit DBCC Event	16	DBCC CHECKIDENT ('testingtableidentity', RESEED...

Public Doc Reference

default trace enabled Server Configuration Option

More Information

The default trace in SQL Server

How good have you found this content?