**A Few Cool Things You Can Identify Using the Default Trace**

**By** [**Greg Larsen**](http://www.databasejournal.com/feedback/http:/www.databasejournal.com/features/mssql/a-few-cool-things-you-can-identify-using-the-default-trace.html)

**FROM:** [**http://www.databasejournal.com/features/mssql/a-few-cool-things-you-can-identify-using-the-default-trace.html**](http://www.databasejournal.com/features/mssql/a-few-cool-things-you-can-identify-using-the-default-trace.html)

If you are running an instance of SQL Server 2005 and above then most likely that instance is running the default trace.  This default trace is a canned Profiler server side trace that automatically starts up when SQL Server starts.  In this article I will explain more about the default trace and show you how to glean some event information from the trace files created by this background trace process.

**What is the Default Trace?**

The default trace is a pre-defined profiler trace definition that comes with the SQL Server installation.  This default trace definition runs as a server side trace and is normally automatically started when SQL Server starts up.  See Table 1, for a list of Profiler events that are captured by the default trace.

|  |  |
| --- | --- |
| **eventid** | **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 |
| 157 | FT:Crawl Aborted |
| 164 | Object:Altered |
| 167 | Database Mirroring State Change |
| 175 | Audit Server Alter Trace Event |
| 218 | Plan Guide Unsuccessful |

**Table 1:  Listing of Profiler events monitored by the default trace**

When you install SQL Server out of the box the default trace definition is set up to run automatically.  This automatic running of the default trace definition is controlled by the "default trace enabled" configuration parameter.  You can review whether or not the default trace definition is setup to automatically start by running the code in listing 1.

SELECT \* FROM sys.configurations

WHERE name like 'default trace enabled'

**Listing 1: Showing the configured settings for the default trace**

When you run the code in Listing 1, if the *value\_in\_use* column is 1 then your trace definition is set up to run automatically when SQL Server starts up.

If your default trace is not running (*value\_in\_use* = 0) then you can configure it to start running when SQL Server starts up by issuing the command in Listing 2.

USE master;

GO

EXEC sp\_configure 'show advanced option', '1';

reconfigure

go

exec sp\_configure 'default trace enabled', 1

reconfigure

GO

**Listing 2: Configuring the default trace to start automatically**

Note that you have to set the advanced option to 1 in order to change the default trace enabled configuration.  If you normally leave this advanced option set to zero you might want to run another sp\_configure statement to set the advanced option back to 0.

The information captured by the default trace is written to a Profiler trace file.  If your default trace file is running you can run the code in Listing 3 to determine where the default trace definition is being written.

SELECT \* FROM sys.traces

WHERE id = 1;

**Listing 3: Displaying information about the default trace file**

The statement in Listing 3 will display a number of different columns of information about the default trace definition.  The *path* column can be used to identify the current trace file that is being written.

**How to View the Information Collected by the Default Trace**

The default trace information is written to a disk file.  There are a number of different ways to read this information.  One method is to pull up the disk file using Profiler.  The other method is to use TSQL code.  Let me explain how to use both of these methods.

The first thing you need to do, regardless of which method you are using, is to identify where the default trace file is being written.  By default the trace files are written to the location where the SQL Server database engine stores it log files, which if you are running SQL Server 2008 R2 using default installation setting, can be found in "C:\Program Files\Microsoft SQL Server\MSSQL10\_50.*<instance name>*\MSSQL\Log", where <instance name> is replaced with the name of your instance.  If your SQL Server is configured to capture a default trace file, another way to determine where the current trace file is being written is to run the code in Listing 4.

SELECT value FROM sys.fn\_trace\_getinfo(1)

WHERE property = 2;

**Listing 4: Displaying log location for the default trace definition**

Once you have the trace file name, you can use the following steps to open the trace file using the SQL Server Profiler GUI interface:

1) Bring Up SQL Server Profiler

2) Select the "File" menu, then in the drop down menu hover over "Open", finally select "Trace File" option.

3) Then either browse to where your trace file exists and select it, or type the full path name to the trace file.  Once the trace file has been selected or entered, click on the "Open" button.

By following these steps your trace file will be opened in the Profiler GUI, where you can now scroll and/or search through all the different events that it captured.

Another way to read the trace file is to use the "::fn\_trace\_gettable" function.  By using this function you can use TSQL to browse your trace file.  You can use the code in Listing 5 to view information in the current default trace file.

DECLARE @filename nvarchar(1000);

-- Get the name of the current default trace

SELECT @filename = cast(value as nvarchar(1000))

FROM ::fn\_trace\_getinfo(default)

WHERE traceid = 1 and property = 2;

-- view current trace file

SELECT \*

FROM ::fn\_trace\_gettable(@filename, default) AS ftg

INNER JOIN sys.trace\_events AS te ON ftg.EventClass = te.trace\_event\_id

ORDER BY ftg.StartTime

**Listing 5: View the current default trace file information**

By viewing the output created when running the code in Listing 5 you can see all the different trace events that have been captured in the current trace file.   I will now expand on the code in Listing 5 and provide you a number of different examples that will show you how to identify different kinds of events that have been captured by the default trace.

**Schema Changes**

The default trace captures an event every time an object is created or deleted, by tracing event\_id's 46 and 47.  You can use these two events to determine when objects are created or deleted and who performed those events.  The code in Listing 6 shows how to select the object created and deleted events from the current default trace file.

DECLARE @filename nvarchar(1000);

-- Get the name of the current default trace

SELECT @filename = cast(value as nvarchar(1000))

FROM ::fn\_trace\_getinfo(default)

WHERE traceid = 1 and property = 2;

-- view current trace file

SELECT \*

FROM ::fn\_trace\_gettable(@filename, default) AS ftg

INNER JOIN sys.trace\_events AS te ON ftg.EventClass = te.trace\_event\_id

WHERE (ftg.EventClass = 46 or ftg.EventClass = 47)

and DatabaseName <> 'tempdb'

and EventSubClass = 0

ORDER BY ftg.StartTime;

**Listing 6: Displaying object CREATE and DELETE events**

By using the code in listing 6 you can identify the objects that were created or deleted in any database other than "tempdb".  You might want to use this method to identify who created and deleted objects in your database.

**Autogrowth Events**

If you have your database files set to Auto-Grow when they run out of space, then each time they grow the default trace file will capture the Auto-Grow event.  By using the code in Listing 7 you can find out every time a database has an Auto-Grow event.

DECLARE @filename nvarchar(1000);

-- Get the name of the current default trace

SELECT @filename = cast(value as nvarchar(1000))

FROM ::fn\_trace\_getinfo(default)

WHERE traceid = 1 and property = 2;

-- Find auto growth events in the current trace file

SELECT

  ftg.StartTime

 ,te.name as EventName

 ,DB\_NAME(ftg.databaseid) AS DatabaseName

 ,ftg.Filename

 ,(ftg.IntegerData\*8)/1024.0 AS GrowthMB

 ,(ftg.duration/1000)as DurMS

FROM ::fn\_trace\_gettable(@filename, default) AS ftg

INNER JOIN sys.trace\_events AS te ON ftg.EventClass = te.trace\_event\_id

WHERE (ftg.EventClass = 92  -- Date File Auto-grow

    OR ftg.EventClass = 93) -- Log File Auto-grow

ORDER BY ftg.StartTime

**Listing 7: Display Auto-Grow events**

The code in Listing 7 takes the information found on Auto-Grow events and provides you some useful information about these Auto-Grow events, like how much your database grew, and how long in milliseconds that Auto-Grow event took.  If you find your databases are having Auto-Grow events occurring frequently then you might want to consider changing your allocation sizes to minimize the number of times an Auto-Grow event occurs.

**Security Changes**

Security Auditors are always wondering when people have been granted or denied permissions to data.  The default trace process captures a number of different security related events.   By using the code in Listing 7 you can identify a number of different security related events, like when logins are added to your servers or are given access to a database, or placed in different server or database level roles.

DECLARE @filename nvarchar(1000);

-- Get the name of the current default trace

SELECT @filename = cast(value as nvarchar(1000))

FROM ::fn\_trace\_getinfo(default)

WHERE traceid = 1 and property = 2;

-- process all trace files

SELECT \*

FROM ::fn\_trace\_gettable(@filename, default) AS ftg

INNER JOIN sys.trace\_events AS te ON ftg.EventClass = te.trace\_event\_id

WHERE ftg.EventClass

    in (102,103,104,105,106,108,109,110,111)

ORDER BY ftg.StartTime

**Listing 8: Showing Security Related Events**

**Summary**

I've only shown you a few different scripts that can be used to view the default trace information.  These samples should provide you with ideas on how to write your own queries to extract information from the default trace file.  There is a great deal of information you can obtain by looking at the default trace data.  Next time you are wondering what might be going on inside your database instance you might consider reviewing the default trace information to help answer your questions.

[**See all articles by Greg Larsen**](http://www.databasejournal.com/article.php/1560691/Gregory-A-Larsen.htm)

[MS SQL Archives](http://www.databasejournal.com/features/mssql/archives/)

**4 Comments**[**(click to add your comment)**](http://www.databasejournal.com/features/mssql/a-few-cool-things-you-can-identify-using-the-default-trace.html#comment_form)

By pl80   April 27 2012 00:35 PDT

There is no mention, that the log file storing the trace is limited to 20MB. Once the file is filled, SQL Server starts another file. Up to 5 files are used. Here's the code (cannot remember where I copied it from) to load info from all 5 of them (and hence see default trace recods much farther back in time): DECLARE @FileName VARCHAR(MAX) SELECT @FileName = SUBSTRING(path, 0, LEN(path)-CHARINDEX('\', REVERSE(path))+1) + '\Log.trc' FROM sys.traces WHERE is\_default = 1; --select @filename SELECT \* into trace\_table FROM sys.fn\_trace\_gettable( @FileName, DEFAULT ) AS gt select \* from trace\_table

[http://assets.devx.com/devx/commentReply2.gif**Reply to this comment**](http://www.databasejournal.com/features/mssql/a-few-cool-things-you-can-identify-using-the-default-trace.html#comment_form)

[http://assets.devx.com/RIARun/alert_big.gif](mailto:commentsupport@quinstreet.com?subject=comment%201330922%20flagged&body=Comment%20flagged%20on%20http://www.databasejournal.com/features/mssql/a-few-cool-things-you-can-identify-using-the-default-trace.html)

By Srinath   April 08 2012 23:36 PDT

Thats handsomely written. Learnt many things.... Big Thank you!

[http://assets.devx.com/devx/commentReply2.gif**Reply to this comment**](http://www.databasejournal.com/features/mssql/a-few-cool-things-you-can-identify-using-the-default-trace.html#comment_form)

[http://assets.devx.com/RIARun/alert_big.gif](mailto:commentsupport@quinstreet.com?subject=comment%202022194%20flagged&body=Comment%20flagged%20on%20http://www.databasejournal.com/features/mssql/a-few-cool-things-you-can-identify-using-the-default-trace.html)

By Steve   April 05 2012 04:03 PDT

Hi Greg Thanks for a great article. There is an extra space in name like 'default trace enabled' BTW These captchas are impossible to read! HTH Steve

[http://assets.devx.com/devx/commentReply2.gif**Reply to this comment**](http://www.databasejournal.com/features/mssql/a-few-cool-things-you-can-identify-using-the-default-trace.html#comment_form)

[http://assets.devx.com/RIARun/alert_big.gif](mailto:commentsupport@quinstreet.com?subject=comment%202009394%20flagged&body=Comment%20flagged%20on%20http://www.databasejournal.com/features/mssql/a-few-cool-things-you-can-identify-using-the-default-trace.html)

By Tiago   April 05 2012 03:34 PDT

I really enjoyed this article. Thanks for your time and for sharing this content. Good work!