**Audit Logons with Extended Events**

[Jason Brimhall](https://www.sqlservercentral.com/author/SQLRNNR), 2020-04-29 (first published: 2020-04-20)

FROM: <https://www.sqlservercentral.com/blogs/audit-logons-with-extended-events-8>

[](http://jasonbrimhall.info/wp-content/uploads/2018/11/db_audit.png)

Some time ago, I wrote an article for SQL Server 2008 to help determine the use of the server since SQL Server 2008 was reaching End Of Life. In that article, I shared a reasonable use of server side trace to capture all of the logon events to the server. Afterall, you have to find out the source of connections and who is using the server if you need to migrate it to a newer SQL Server version. You can read that article [here](https://jasonbrimhall.info/2019/07/17/life-support-2008-audit-logons/).

Soon after, from various sources, I received requests on how to perform a logon audit using the more preferred, robust, venerable, awesome tool called Extended Events (XEvents). In response, I would share a login audit session to each person. In this article, I will share my login audit solution and give a brief explanation. I use a solution like this on more than 90% of my client servers and I find it highly useful.

Auditing Logons

The Events to capture SQL Server logon/logoff activities were not a part of the original release of XEvents in 2008. The requisite events did not become available until SQL Server 2012. Why mention SQL Server 2008 at all given it has reached its End of Life you may ask? Well, as luck would have it, there are still plenty of 2008/R2 instances out there in the world that are yet to be upgraded still. So, it is useful to continually point it out. This session, and this information, does NOT apply to anything prior to SQL Server 2012.

The events in XEvents that we need happen to be called: sqlserver.login and sqlserver.logout. Here is a little more info on those events along with a query that can retrieve the details being shown here.

SELECT xp.name AS PackageName,xo.name AS EventName, xo.description --, xo.capabilities\_desc

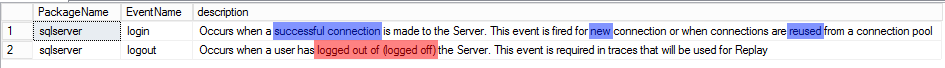
FROM sys.dm\_xe\_objects xo

INNER JOIN sys.dm\_xe\_packages xp

ON xo.package\_guid = xp.guid

WHERE xo.name = 'login'

OR xo.name = 'logout';

[](http://jasonbrimhall.info/wp-content/uploads/2020/04/xe_logonaudit.png)

As shown in the preceding image, there are two events of interest that are needed for auditing logon events – sort of. These events are only useful to capture the successful connection (or connection pooling reuse connections) and the logoff events. To take this a step further and capture the failed logins, we can also use an Event that was available in 2008, error\_reported. The use of the logout event would be an optional event in this case but could be of use during the troubleshooting under certain circumstances.

Focusing on the login and error\_reported events, I have two very useful XEvent Sessions I like to use for my client servers. First part of the audit logon solution is the audit of successful logins with this XEvent Session.

USE master;

GO

-- Create the Event Session

IF EXISTS

(

SELECT

\*

FROM sys.server\_event\_sessions

WHERE name = 'SVRLoginAudit'

)

DROP EVENT SESSION SVRLoginAudit ON SERVER;

GO

EXECUTE xp\_create\_subdir 'C:DatabaseXE';

GO

CREATE EVENT SESSION SVRLoginAudit

ON SERVER

ADD EVENT sqlserver.login

(SET

collect\_database\_name = (1)

, collect\_options\_text = (1)

ACTION

(

sqlserver.sql\_text

, sqlserver.nt\_username

, sqlserver.server\_principal\_name

, sqlserver.client\_hostname

, package0.collect\_system\_time

, package0.event\_sequence

, sqlserver.database\_id

, sqlserver.database\_name

, sqlserver.username

, sqlserver.session\_nt\_username

, sqlserver.client\_app\_name

, sqlserver.session\_id

, sqlserver.context\_info

, sqlserver.client\_connection\_id

)

)

ADD TARGET package0.event\_file

(SET filename = N'C:DatabaseXESVRLoginAudit.xel', max\_file\_size = (5120), max\_rollover\_files = (4))

WITH

(

STARTUP\_STATE = OFF

, TRACK\_CAUSALITY = ON

);

/\* start the session \*/

ALTER EVENT SESSION SVRLoginAudit ON SERVER STATE = START;

GO

Next, I like to use a separate session for the failed logins. This makes it easy to keep the audit files separate and only share the necessary files to others (e.g. only the failed logins instead of giving them both successful and failed which may be way too much information). In addition, I like this method because it makes searching the audit files easier and more focused.

USE master;

GO

-- Create the Event Session

IF EXISTS ( SELECT \*

FROM sys.server\_event\_sessions

WHERE name = 'Audit\_FailedLogon' )

DROP EVENT SESSION Audit\_FailedLogon

ON SERVER;

GO

EXECUTE xp\_create\_subdir 'C:DatabaseXE';

GO

CREATE EVENT SESSION [Audit\_FailedLogon] ON SERVER

ADD EVENT sqlserver.error\_reported (

ACTION (

package0.event\_sequence

, sqlserver.client\_app\_name

, sqlserver.client\_hostname

, sqlserver.client\_pid

, sqlserver.nt\_username

, sqlserver.server\_principal\_name

, sqlserver.session\_nt\_username

, sqlserver.transaction\_sequence

, sqlserver.username

)

WHERE ([severity]=(14) AND [error\_number]=(18456) AND [state]>(1)) )

ADD TARGET package0.event\_file

(SET filename = N'C:DatabaseXEAudit\_FailedLogon.xel'

)

WITH ( MAX\_MEMORY = 4096 KB

,EVENT\_RETENTION\_MODE = ALLOW\_SINGLE\_EVENT\_LOSS

, MAX\_DISPATCH\_LATENCY = 3 SECONDS

, MAX\_EVENT\_SIZE = 0 KB

, MEMORY\_PARTITION\_MODE = NONE

, TRACK\_CAUSALITY = ON

, STARTUP\_STATE = ON

);

In this second session, I have the predicate defined such that I will only get the failed login events. Funny thing about this, I can often catch 3rd party vendors trying to login to client systems in very odd ways (including putting the password into the user name box which in turn causes the password to be stored in clear text in the error log).

Could these sessions be combined into a single all-purpose session? Sure! That is up to your discretion. I prefer the separate sessions for ease of management and review.

The Extra Mile

Let’s say we wanted to go ahead and add the logout event to our session, we can then proceed with a slight modification to the successful login session so that it would look something like the following.

USE master;

GO

-- Create the Event Session

IF EXISTS

(

SELECT

\*

FROM sys.server\_event\_sessions

WHERE name = 'SVRLoginOutAudit'

)

DROP EVENT SESSION SVRLoginOutAudit ON SERVER;

GO

EXECUTE xp\_create\_subdir 'C:DatabaseXE';

GO

CREATE EVENT SESSION SVRLoginOutAudit

ON SERVER

ADD EVENT sqlserver.login

(SET

collect\_database\_name = (1)

, collect\_options\_text = (1)

ACTION

(

sqlserver.sql\_text

, sqlserver.nt\_username

, sqlserver.server\_principal\_name

, sqlserver.client\_hostname

, package0.collect\_system\_time

, package0.event\_sequence

, sqlserver.database\_id

, sqlserver.database\_name

, sqlserver.username

, sqlserver.session\_nt\_username

, sqlserver.client\_app\_name

, sqlserver.session\_id

, sqlserver.context\_info

, sqlserver.client\_connection\_id

)

),

ADD EVENT sqlserver.logout

(

ACTION

(

sqlserver.sql\_text

, sqlserver.nt\_username

, sqlserver.server\_principal\_name

, sqlserver.client\_hostname

, package0.collect\_system\_time

, package0.event\_sequence

, sqlserver.database\_id

, sqlserver.database\_name

, sqlserver.username

, sqlserver.session\_nt\_username

, sqlserver.client\_app\_name

, sqlserver.session\_id

, sqlserver.context\_info

, sqlserver.client\_connection\_id

)

)

ADD TARGET package0.event\_file

(SET filename = N'C:DatabaseXESVRLoginAudit.xel', max\_file\_size = (5120), max\_rollover\_files = (4))

WITH

(

STARTUP\_STATE = OFF

, TRACK\_CAUSALITY = ON

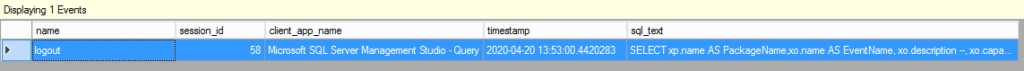
);

/\* start the session \*/

ALTER EVENT SESSION SVRLoginOutAudit ON SERVER STATE = START;

GO

And, to show what might occur with this session, I can see data similar to this for the logout event.

[](http://jasonbrimhall.info/wp-content/uploads/2020/04/logout.png)See how easy that is!

[](http://jasonbrimhall.info/wp-content/uploads/2018/03/xe_flysuperhero.png)The Wrap

This article has shown how to audit the logon events for SQL Server 2012 and beyond through the use of XEvents. It also happens to be an excellent follow up to another recent article – [here](https://jasonbrimhall.info/2019/07/03/is-that-database-in-use/). An essential function of any DBA is to have a clear understanding of the activity (logons) that is occurring on the servers. This audit session will help you do exactly that!

Through the power of XEvents, we can accomplish a great many things and reach extensive insights into our database footprint. This post is just one of many in the XE Series, of which you can read more – [here](https://jasonbrimhall.info/2015/09/08/learning-extended-events-in-60-days/).

In addition, this post is similar to many in the back to basics series. Other topics in the series include (but are not limited to): [Backups](http://bit.ly/2heI2eG), [backup history](http://bit.ly/2h7Dpmr) and [user logins](http://jasonbrimhall.info/2016/12/23/user-contains-invalid-characters-back-to-basics/).

https://jasonbrimhall.info/?feed-stats-post-id=6394

Related Posts:

* [Life Support 2008 - Audit Logons](https://jasonbrimhall.info/2019/07/17/life-support-2008-audit-logons/) July 17, 2019
* [Is That Database in Use?](https://jasonbrimhall.info/2019/07/03/is-that-database-in-use/) July 3, 2019
* [Easy Permissions Audit](https://jasonbrimhall.info/2019/01/21/easy-permissions-audit/) January 21, 2019
* [Easy Audit Index Changes](https://jasonbrimhall.info/2019/12/27/easy-audit-index-changes/) December 27, 2019
* [Monitor Database Offline Events](https://jasonbrimhall.info/2018/11/15/monitor-database-offline-events/) November 15, 2018