**[Auditing Data Access in SQL Server for GDPR Compliance](http://www.sqlservercentral.com/blogs/matthew-mcgiffen-dba/2017/09/26/auditing-data-access-in-sql-server-for-gdpr-compliance/)**

Posted on 26 September 2017

FROM: <http://www.sqlservercentral.com/blogs/matthew-mcgiffen-dba/2017/09/26/auditing-data-access-in-sql-server-for-gdpr-compliance/?utm_source=SSC&utm_medium=pubemail>

In relation to the GDPR, I’ve recently been looking at the tools available within SQL Server to support auditing our databases. Many of you out there will already has full-scale auditing solutions in place. For others though, the prompting of GDPR could be the first time you’re really thinking about it.

This post is a brief overview and comparison of the SQL features that might help you out, with some suggestions of what you might want to think about in terms of auditing.

The GDPR has clear requirements that we must be able to provide detail of changes to personal data. That means being able to say who changed an item of data, when they changed it, and what was the before and after value.

The regulation also states that we must ensure that data is only accessed by appropriate parties, and for the explicit purposes that the person the data relates to has consented. We may feel that this latter requirement is satisfied by our security processes, but in practice, if we wish to prove this then we may also want to audit all data access – including when data is simply read but not changed.

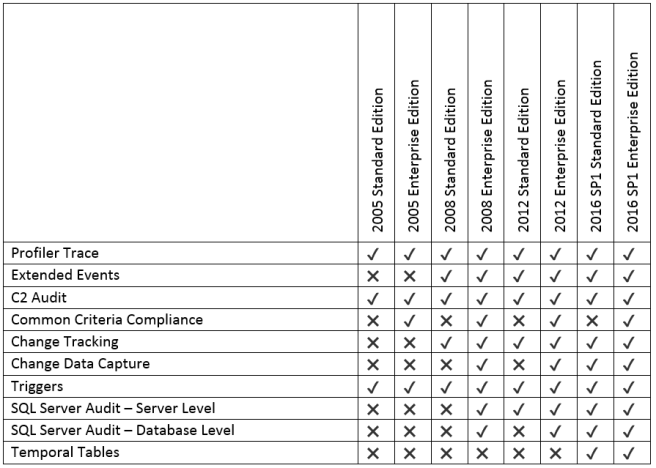
For auditing we need to consider all personal data held, whether that is data provided by an individual or data about an individual that supplements this. Auditing should therefore not be limited to the definition of “personal” within GDPR, namely Personally Identifiable Information (PII). We may also, for our own purposes outside of the GDPR requirement, wish to audit other information related to our business process – for instance where approvals are required.

In terms of individuals accessing or changing data there are several contexts of which we need to be aware:  
Application users

* Support access via specific tooling
* Support access direct to the database
* Illegitimate access via an application or tooling
* Illegitimate access direct to the database

**Summary of Tools available within SQL Server**

The table below lists the technologies we will look at as well as in what versions/editions of SQL Server they are available. Some versions are omitted where the availability of features is the same as in the previous version:



**Profiler Trace**

SQL Server traces allow you to capture information about all access to the database/server. As such this technology is sometimes used as part of an auditing solution – in fact both the C2 Auditing and Common Compliance Criteria features utilise it in one way or another.

It is more useful, however, for capturing instance level security events (such as Logons and failed Logins) than for monitoring changes to individual data items.

There can also be quite an overhead to running traces, though this can be minimised if they are created as server-side traces

**Extended Events**

Introduced in SQL 2008, Extended Events were intended as a lightweight replacement for Profiler Traces. Still, however, they would be more appropriate for capturing security events than for monitoring data access and change – although SQL Server Audit which we’ll look at later is implemented via Extended Events.

**C2 Audit**

This feature is supported in all versions and editions of SQL Server up to 2017, though it has been marked as deprecated for some time.  
C2 was a rating established by the US Department for Defence applicable to security levels of computer systems, mainly focusing on access to resources. The C2 Audit within SQL server is implemented via server-side traces and captures events such as:

* Server shutdowns and restarts
* Successful and failed logins
* Successful and failed use of permissions when accessing database objects
* Successful and failed use of permissions when running any command against the database server

It can be seen from the events captured that while this sort of data capture may be useful as part of an overall audit strategy, it won’t cover the audit of individual data required by the GDPR.

If implementing C2 Auditing on an instance, it is important to be aware that it will have some performance impact on a server as each event must be written to the audit file before the operation can complete. In fact, if the audit file cannot be accessed (or disk space runs out) then SQL Server will stop completely until that is resolved. However, it is a limited set of items being audited so the performance impact in general should not be too considerable.

**Common Criteria Compliance**

Available from SQL 2005 SP1 Enterprise, and in standard edition from SQL 2008, Common Compliance Criteria is described by Microsoft as the replacement for C2 Audit.

Enabling this option configures a couple of other behaviours within SQL server to comply with the “Common Criteria for Information Technology Evaluation” International Standard.

From an auditing point of view, it enables a level of login auditing which can be viewed by querying sys.dm\_exec\_sessions for currently connected users.

This information includes:

* Last Successful login time
* Last unsuccessful login time
* Number of Unsuccessful login attempts since last successful login

**Triggers**

Available in all editions/versions from SQL 2005. Triggers were also available before that, but additional events were added in 2005.

It is a common usage of Triggers to perform some level of database auditing, for instance to capture a modified datetime for a record, or even the identity of the logged-on user. They can also be used to capture more information about changes – for instance to maintain Audit tables as data is inserted, updated or deleted.

The main potential disadvantage of triggers for Auditing is performance. Technologies designed specifically for the purpose may be more lightweight.

**Change Tracking (CT)**

Available in SQL Server 2008 standard edition onward, CT is not designed to be used for Audit. Rather it is intended to support ETL processes by identifying changes to tables.

When enabled Change Tracking simply identifies that a given row has been inserted, updated or deleted. It doesn’t capture the content of the change, for instance that a given column was changed from value ‘A’ to value ‘B’. It doesn’t maintain a change history, so if I insert a record, update it five times, then delete it, when I check change tracking it will only tell me that it was deleted.

**Change Data Capture (CDC)**

Available in SQL Server 2008 Enterprise Edition onward, from 2016 SP1 CDC is also available in Standard Edition. Like CT, CDC is not intended as an audit technology – however due to the level of information captured it can be useful in that context.

As well as identifying that data has changed, CDC captures the actual changes and maintains the history in a separate table. Unlike a similar hand-cranked solution using triggers, it manages this asynchronously so has only light performance overhead.

One of the key things missing from CDC from an Audit perspective is that while it captures the data change, it doesn’t also record the Login identity making the change or the datetime for the change.

A Connect item was raised to change this, however the Microsoft response was to reject by design as CDC “is not designed to be an auditing feature”. Microsoft’s recommendation is to use SQL Server Audit instead:

<https://connect.microsoft.com/SQLServer/feedback/details/283707/cdc-options-to-capture-more-data-username-date-time-etc>  
Despite this, if the table itself has columns to capture the user identity and the modified datetime, and the application ensures these are set, then these columns will be included in the change data capture in which case a meaningful Audit of changes to data is provided.

CDC can be enabled on a per table basis, and you can configure which security role(s) have access to the change log. You can also define the retention period which avoids you having to implement a separate solution for cleaning up old audit data.

One thing to bear in mind is that due to the asynchronous process involved to log changes (which uses SQL Server Agent) there is no transactional guarantee that changes will reach the capture log, though the risk of data loss is likely to be minimal.

**SQL Server Audit**

Available in SQL Server 2008 Enterprise edition, Server level auditing came to Standard edition in 2012 and all features are available in all editions from 2016 SP1.

SQL Server Audit is the out of the box feature that is intended for the purposes of Audit, so logically we may think it is the best one for us to be using. Certainly, it is what Microsoft would recommend for auditing within the database.

Auditing is implemented using Extended Events and there are many Audit events you can enable it to capture. Events are either logged to a file, or to the Windows Security log where it is more difficult to tamper with them. You can then either view the events directly in that destination or through a GUI within SSMS.

Depending on the scope of the events monitored, you can choose to Audit either at an object level (for instance table by table) or at schema or database level.  
In the context of GDPR we may choose to Audit data access – insert, select, update and delete. What this sort of Audit will give us is not, as we might expect, the ability to view who accessed or changed specific data. Rather it captures the queries being executed against the objects monitored.

As such it’s probably not the best method for being able to track a change history for a piece of data, though that could be done in theory by analysing the individual queries. Equally, if we want to know who read a specific record, this would require analysis of the queries executed.

Because of that, while SQL Server Audit may form an important part of an overall Audit strategy, it may not, on its own, give us what we need.

It does however allow us to capture all queries that have, for instance, touched certain tables. As such it is the only technology available to use within SQL Server that may allow us to monitor for abuse of privileges. For instance, I may be the DBA and require admin rights over the SQL Server instance, with SQL Server Audit the queries I execute against tables containing personal information would be captured and any dubious querying patterns could be identified.

An important part of the use of SQL Server Audit is going to be proactively monitoring the logs to make sure data access is being carried out appropriately.

**Temporal Tables**

Temporal tables were introduced in SQL Server 2016 and are a way of easily tracking changes to data – as well as being able to easily query what the value was at a given point in time.

Like CDC they do not capture the user context making the data change, so this must be handled by additional application logic if we require it.

The logging of change data for temporal tables is a synchronous operation performed as part of the transaction that changes data in the main table. Microsoft documentation however states that this creates negligible overhead for inserts, and while there is greater overhead for updates and deletes this is still significantly less than a custom solution (for instance using triggers).

Temporal tables work by having two user tables for each table being tracked. One is the original table – known as the current table, and the other is the history table which includes start and end times for which a given version of the data was current. New commands for querying the state of data at a given point in time make this a seamless operation.

**Application Audit**

One issue with any auditing that we perform at the database level is that any well designed application, from a security context, is likely to be accessing the database via a service account, rather than application users being granted direct database access. As such it is not possible for pure database auditing to capture a meaningful user context where operations are conducted through the application.

For instance, it may be quite legitimate from the application for a user to update details relating to a customer. We need to maintain a record of those changes so we require the application to log the user context – usually into an “UpdatedBy” column.

There may also be the case where semi-structured, or unstructured data is stored in blob forms, such as XML, JSON or other document formats. In these cases, a database level audit could capture the before and after states of the data, but it would require additional analysis to identify what actual elements of the data within the document had changed. If we want to provide an easily digestible change history then we need to consider implementing this at an application or tooling level.

Finally, we may want to record user access and activities from the application. For instance, the date and time of connections as well as what was accessed.

**What might a best practice SQL Data Access Audit solution look like?**

Which (if any) of these technologies we use is going to depend very much on the versions and editions of SQL Server we have installed. In an ideal world, to make our lives easy we would all have Enterprise edition – or be on SQL 2016 SP1 or later – and have an extensive choice. In some cases, it may be that the access to additional security and audit components becomes the compelling reason for choosing Enterprise over Standard edition, or for upgrading to a later version of SQL Server.

**C2 or Common Compliance**

It makes sense to implement one of these, ideally Common Compliance where this is available, as this gives us increased security within SQL Server at little cost and minimal effort to implement. Common Compliance is preferable both as C2 is deprecated and that C2 poorly managed (losing disk access) can create additional risks to application availability. Both offer additional levels of logging connections to our SQL Server boxes that may be useful in identifying those logging onto the server for nefarious purposes.

**Change Capture**

In particular for GDPR we will want to make sure we have some form of capture and logging of changes to personal data. We have a choice of technologies here.

**Triggers**

Triggers can provide a serviceable solution but the performance overhead is high so it may be better to use a built-in technology where one is available.

**CDC**

CDC provides a low overhead tool for managing change capture. However, it is only available in Enterprise Edition until SQL 2016 SP1.

**Temporal Tables**

These are the latest and greatest, so it may make sense to use where we are using SQL Server 2016, although as it is a synchronous operation it may have more of an immediate performance impact than CDC.

I hope to look at a performance comparison between these different methods of data capture in a later post.

**Application Auditing**

Whatever tool is used for change capture, application auditing is required to make sure the relevant user context is captured. It’s also worth considering logging user activity.

**SQL Server Audit**

For a best practice solution, where this is available to us it makes sense to use it. We can use it to capture data access even though the format is not going to be the most easy to analyse . We should also use it to capture events that could relate to anyone making changes to our Change Capture. If anyone makes changes to the Audit itself then this gets captured by the audit so we can detect tampering. We should make sure the Audit is saved to the Security log then that is also tamperproof. We can’t guarantee that we will capture all events, but we can reasonably guarantee that we will detect it if someone prevents us from doing so (e.g. turning off Auditing or Change Capture).

**Conclusions**

There are a number of components you may want to implement in order to deploy an effective data access auditing solution. In some cases your choices may be limited by the versions and editions of SQL your platforms currently sit on.  
In some cases though, where the version and edition of SQL we use supports these technologies, then there is not a great deal of effort required to implement them.

In subsequent posts I’ll hope to show some simple examples in practice, as well as look briefly at the impacts on server performance.

[http://feeds.wordpress.com/1.0/comments/matthewmcgiffendba.wordpress.com/1074/](http://feeds.wordpress.com/1.0/gocomments/matthewmcgiffendba.wordpress.com/1074/)https://pixel.wp.com/b.gif?host=matthewmcgiffen.com&blog=124597259&post=1074&subd=matthewmcgiffendba&ref=&feed=1

[](http://www.sqlservercentral.com/blogs/matthew-mcgiffen-dba/)

[**Matthew McGiffen DBA**](http://www.sqlservercentral.com/blogs/matthew-mcgiffen-dba/)

I started my career in IT as a developer with Visual Basic, C# and ASP working with SQL Server backends. For the last ten years though I’ve focused on the SQL Server side of things. Currently I’m working for a leading software vendor for the financial industry as a DBA, doing a mixture of development and data architecture. If I have a speciality it is in making things run faster.

**Comments**

[Leave a comment on the original post [matthewmcgiffen.com, opens in a new window]](https://matthewmcgiffen.com/2017/09/26/auditing-data-access-in-sql-server-for-gdpr-compliance/#comments)

http://www.sqlservercentral.com/Resources/Images/loading.gifLoading comments...

**Posted by Erik on 27 September 2017**

Thank you for this great list of auditing techniques. Although, it would be very helpful to share the encryption possibilities GDPR requires as lots of us are still using SQL 2008/2012 where always encrypted is not an option. I tried to google some information about encrypting sensitive data with the least impact on performance, but with no luck yet.  
Best Regards.  
Erik

[Like](https://matthewmcgiffen.com/2017/09/26/auditing-data-access-in-sql-server-for-gdpr-compliance/?like_comment=205&_wpnonce=5601a08a16)Like

**Posted by Matthew McGiffen on 27 September 2017**

Thanks Erik – that’s a good suggestion, I might do a post on that. In the meantime there’s a short article on SQLMAG I found: <http://sqlmag.com/database-security/sql-server-encryption-options>

[Like](https://matthewmcgiffen.com/2017/09/26/auditing-data-access-in-sql-server-for-gdpr-compliance/?like_comment=206&_wpnonce=3c5fa119ff)Like

**Posted by Setting up Change Data Capture (CDC) &#124; Matthew McGiffen DBA on 3 October 2017**

[…] mentioned in my post Auditing Data Access in SQL Server for GDPR Compliance CDC can be a useful tool for capturing a change history for specific data – which is something […]

[Like](https://matthewmcgiffen.com/2017/09/26/auditing-data-access-in-sql-server-for-gdpr-compliance/?like_comment=216&_wpnonce=71a05d303d)Like