07-28-2021

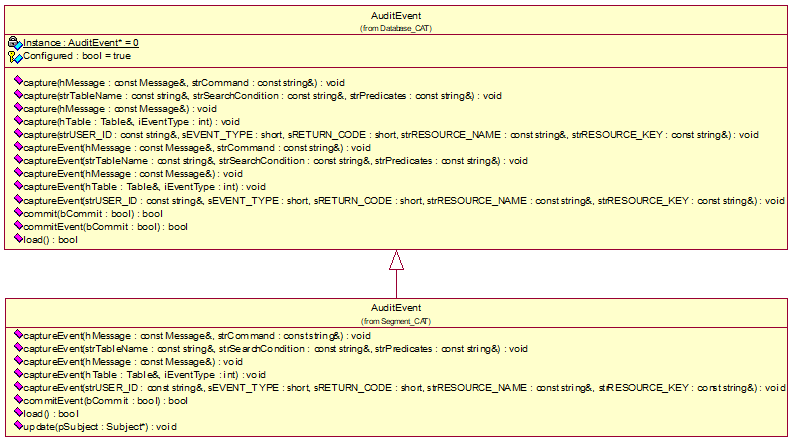
**How auditing works in the Data Navigator server**

DN Tasks that perform operations that require an audit trail include the following statement in their initialize method:

new segment::AuditEvent

The following DN tasks perform this initialization: AC,AM,CQ,EM,EQ,FM,IM,LM,OC,WM(dll),XC,XT

This establishes segment::AuditEvent as the audit event handler. segment::AuditEvent (BS27) is derived from database::AuditEvent (DB44)



database::AuditEvent has 5 static capture methods that can be called to capture messages, commands, search conditions, table changes, logon attempts and other resources and operations. Calls to these static capture methods are re-routed to 5 associated virtual captureEvent methods.

database::AuditEvent is a form of singleton. It maintains a static instance pointer (which is set by calling new segment::AuditEvent. database::AuditEvent routes the static calls to virtual methods using the static instance pointer. For example: AuditEvent::capture(…) becomes m\_pInstance->captureEvent(…) .

segment::AuditEvent stores events in a container of AuditEventSegments (BS28). All events in the container are written to the AUDIT\_EVENT table anytime Database::commit() is called. Each of the database implementations: OracleDatabase, PostgreSQLDatabase, ODBCDatabase, and DB2Database have a call to database::AuditEvent::comit() in their commit methods. The event container is emptied after being written to the database.

Audit events are written from the AUDIT\_EVENT table to a XML audit log file by AuditEventFile (MG28). See “XML Format for Centralized Logging.pdf”. AuditEventFile is an ExportFile class that is run by the DF task. The LM tasks accepts a “RESET AUDIT” command from the Event Manager task. The frequency of when EV sends this command to LM dictates how often AUDIT\_EVENT table rows are written to the DX\_DATA\_YYYYMM tables by the AuditEventFile. Upon receiving the “RESET AUDIT” command from the Event Manager or the DN Console, LM issues m\_pAuditEventFile->createFile(). This causes AuditEventFile to schedule itself in the DX\_DATA\_CONTROL table so that the DF executes it. Once the XML file has been produced in the DX\_DATA\_YYYYMMDD table, the DT task will write it to disk based on DFILES definition in its extract file.

PA-DSS requirements state that audit records must be made available no longer than every two hours.

**Auditing of static files:**

Auditing of executable files and configuration files is performed by the FS task. The FS tasks submits a PERL script from CXOXSCAN.txt every 10 minutes. The 1st time the perl script runs it will create an MD5 hash for each file in Alpha/Bin and will save that in the bin directory it is running from as alpha\_bin.txt. It will also do the same for Alpha/Source saving as alpha\_source.txt and for Pprod saving as pprod.txt.

It will also write out an INSERT alert for each file in these directories because they will appear to be new files. The alerts are written to NODE001/CUSTQUAL/Pending/alerts\_YYMMDDHHMM00.txt (where MM will be 00,10,20,30,40,50).

Every minute the FS task (if it is not on a 10 minute boundary where the perl script is submitted) will read any and all alert files that are in the Pending folder and will insert them into the AUDIT\_EVENT table. It will move those files to the Complete folder.

The PERL script submitted by FS also writes to a daily log file located in NODE001/CUSTQUAL/Pending/log\_YYMMDD.txt

All alerts writted to AUDIT\_EVENT table are then picked up and written to the XML log file as previously described in this document.