## 

[Introduction](#_fchp502hhkqw)

[Why are we implementing auditing?](#_navg4hmmn1k)

[What are the limitations?](#_gygx4t3jgnsu)

[Why the DB2 12.1 upgrade is used as the solution for Auditing](#_ptdp3sgwo02c)

[Why an upgrade?](#_ngkttwfmx2e2)

[Minimum Software Requirements](#_m62j8udzrzr5)

[Important note before starting.](#_ln14s0mn8bh0)

[Setting up policies for India regulations:](#_pqswlq9llame)

[Activate Auditing](#_kqkzxk7vr857)

[For the instance:](#_aqkug6e5lp51)

[For the database:](#_4gyr1xh2hq9e)

[Stopping Auditing](#_vn24ttsrfbw4)

[For the instance](#_hamu28akaw18)

[For the database](#_aixt57qse03)

[Dropping and removing Objects from an Audit Policy](#_a6dr3acf4kjk)

[For tables:](#_5mdb1kufeill)

[For Users](#_wjb5czd4sx3x)

[Removing the database user from the database policy](#_x93lyjap3gj3)

[Removing an Audit Exception for Trusted Context](#_cce3lo20hbv0)

[Queries for auditing](#_w50ay34m2wt2)

[SYSCAT.AUDITUSE](#_bojjz1jwuqn9)

[SYSCAT.AUDITPOLICIES](#_3wr00o4fm1hb)

[Appendix](#_4utylsdbzzr6)

[DB2 Upgrade to Version 12.1](#_exsh5waox6g)

[Steps for Auditing](#_jhxww4wmsskf)

[1. Instance level auditing](#_2chjymvpk7fv)

[2. Database level (policies)](#_akx2tofsfgff)

[Testing that auditing works.](#_hfblgktivabl)

[Extract Archived Logs to Readable Format](#_e55qtlj8p3e1)

[Query Loaded Tables.](#_vf88uow43v8s)

[Troubleshooting, Assorted commands and Notes](#_rv56af6x1b2y)

[Assorted commands - Reference](#_kv15gz91q27i)

[Create or adjust the AUDIT buffer in the DBM](#_u1tty9r8gr6u)

[Configure the data path and the archive path.](#_sguif4rsz81e)

[The AUD0000I Operation succeeded.  
  
Start or stop instance level auditing.](#_dpf0gz5xt110)

[Show instance level auditing](#_tphx8xnbtozj)

[Create an audit policy for the database auditing.](#_d29ldohb8wqf)

[Flush audit buffers - (before archiving active logs)](#_mj1q9tddgqmr)

[Move Active Audit Files to Archive](#_8ivjouv0aoo)

[Extract Archived Audit Log Files to Comma-delimited Files](#_e2jw2d9nd30h)

[Loading files to tables](#_lqk698v7xof1)

[SQL Queries for Audit Tables](#_l3srma5nofk2)

[SAP, IBM and other resources](#_ise8tsgw2w9s)

[Db2audit shows unexpected SYSTOOLS.HMON\_ATM\_INFO data](#_93u61q7x4bmi)

[OSS Note 1474404 - DB6: Poor database performance with DB2AUDIT](#_hf451utup702)

[Load Pending State](#_6pvgtpavis9w)

[How to reset LOAD PENDING state](#_ps33p3t5admj)

[db2audit extract returns AUD0036N](#_qrjruv98boaj)

[Helpful sites:](#_h63l2da5ijgy)

[Using db2audit with SAP Applications - Website and Tutorial](#_los95m785cl6)

[OSS Notes](#_yx9x15mkgkfj)

[1474404 - DB6: Poor database performance with DB2AUDIT](#_9k1yv9xomfqf)

## 

## Introduction

This is intended for the CAP system and is DB2 specific. It can be adjusted for SAP systems running DB2 or for DB2 systems in terms of auditing at the operating system level. The table auditing described at SAP level can be adapted for any SAP ABAP-based system. We will use CAI in this exercise since CAP is not yet upgraded.  
  
For reasons explained later, we are doing auditing at the operating system level (and possibly at the SAP level) for 52 selected tables . Most of this is due to limitations of DB2 native auditing and the realities of a busy OLTP system. This also has an additional policy which runs concurrently against a role granted to DBAs. Thus the auditing of both policies are combined into one file for the instance and one for each database audited. (There is only one database in this case.)

What is covered in this document?

As far as implementation, this document does not cover setting up the creation of the AUDIT filesystem. Nor does it cover the audit profile that exists already for role-based auditing. This document is already long enough.

You can find that material here [Setting up DB2 Audit in Sox Systems](https://docs.google.com/document/d/1bnZrru67WzTjtJtIQcqEYKKB1drbOuCyrVGOoZZ4KJs/edit?tab=t.0#heading=h.s67846r224ou)

### Why are we implementing auditing?

There is an auditing requirement for India for all DML by the database schema, in this case SAPCAP, for a selected set of tables. DB2 Auditing doesn’t quite work that way, it picks up the select statements too, so there are a number of steps to accomplish this goal and ensure the system isn’t overwhelmed. There’s also a requirement for the archived auditing files to be housed at a server located in India for legal reasons. This requires archiving and transferring archived files and possibly any other data required. As the requirements have not been formally defined as of this writing, the requirements are going to be a bit generic and undefined.

### What are the limitations?

### 

DB2 auditing doesn’t allow you to **only** look for records that are changed, it also pulls in any SELECT statements generated in the SAP system. This is quickly 20 - 30GB a day of data written out to files on a test system with minimal activity. Clearly we want the recording to be limited to just INSERT, DELETE or UPDATE changes which isn’t possible so far. A better compromise is to include SAP application and DB2 level auditing. It’s also picking up commands issued by the SYSTOOLS schema and other operations like tp and R3trans from SAP that are executed at the operating system level. It will also pick up IBM and DB2 specific processes and schema like SYSTOOLS.

So what do I mean when talking about SAP application level auditing? At the time of this writing, you are left with third party solutions or SAP table auditing which already exists on the system and provides a comprehensive view of the changes. This is managed per table at the data dictionary level via transaction code SE13 to allow logging of changes at the SAP application level. To put a finer point on this, any changes that come from an SAP work process of any kind are covered here.   
  
Given a busy system like SAP, you can’t really monitor the schema user at operating system level if every SELECT statement at the SAP level is logged. The volume of files would be prohibitive. Imagine someone running an SE16 refresh on table count or a routine that reads from a driver table and selects a single record each time from another table.   
  
The solution is to log the changes made by the application’s work processes into SAP native auditing, this approach records only the changes to the table from the SAP work processes to a table in SAP.

The remaining changes to the tables from the operating system users such as the instance and schema owners should be minimal but will also reflect SELECT statements. This will be written out to physical files at the operating system. The desire is to keep this output as small, yet accurate, as possible.

These can be filtered out at some point with the reporting from the DB2 native auditing but only after it’s recorded.  
  
To accomplish the DB2 level native auditing to function as requested with minimal “noise”, SAP has created a DB2 level auditing exception called a “trusted context”. This requires DB2 version 12.1 or higher.

The end effect is that the changes made at SAP level can be accessed via SAP native tools (and even extracted) and the operating system user who made changes to the tables can be tracked in audit files that can be converted to files that are human readable or to database tables.

The SAP Note provides information on how to use IBM Db2 Version 12.1 for Linux, UNIX, and Windows (in the following referred to as Db2 12.1) with SAP applications based on SAP Application Server ABAP or Application Server Java. We are going to focus on the DB2 Auditing facility upgrades here.

<https://me.sap.com/notes/3520953>

### Why the DB2 12.1 upgrade is used as the solution for Auditing

Let’s go a little further into the requirements as I understand them.

One of the requirements is to record modifications to any data by the SAP schema user for the CAP schema (SAPCAP) user. Before DB2 12.1, DB2 Auditing was severely limited in previous versions in that it will audit insert, update and delete and select. But more importantly from an SAP perspective it also audits the incredible amount of select statements on the audited tables with no way to exclude these from being written out. This results in potentially terabytes of data per week at the filesystem level which will have to be stored as well as an incredible amount of resources to write and archive. Since the auditing can never be interrupted, there’s monitoring on various resources such as filesystems and other considerations that would need to be implemented to prevent outages. This is a lot of work and planning to get right and ensure auditing does not have gaps or missing data with a high degree of risk or even worse, halt the system.   
  
There’s also a request regarding a requirement that the changes made to a row, either before or after, will not be shown in the DB2 Auditing Facility, just the SQL statement and (depending on audit settings) the data used in the statement. This is indeed the case and byIBM’s design. SAP auditing (table logging) goes a little bit further and will show changes to fields. It will not show before and after images for updates either.   
  
While you can take it to that extreme and produce images of a row with before and after a change through triggers, it’s potentially an issue for upgrades and patching and a question of volume.

This version brings a much needed enhancement to the DB2 Auditing Facility. What this allows is to let SAP handle the auditing of statements generated by its own work process connections to the database via the SAP schema user to be excluded by the DB2 Auditing facility through the “trusted context” feature. Auditing at that level will instead be handled via the well established native table logging for the work processes without the overhead of millions of SELECT statements in DB2.

This is particularly useful to reduce the amount of overhead and resources required and handles the storage and resources for that. There’s also built in reporting and archiving with a great deal of the tracking and auditing metadata already included. The DB2 Auditing Facility requires a fair bit of custom reporting and automation as well which is dependent upon requirements.

In this case we are auditing the SAP schema user which is used by the SAP work processes and can also be used at an operating system level. The amount of usage by a user or process at the operating system level should be minimal and thus easily manageable.   
  
There are other actions that need to be taken with the upgrade to a new version of the database that are mentioned later in this document.

### Why an upgrade?

<https://help.sap.com/doc/7367f81b468e4480b3c550669b3534aa/CURRENT_VERSION/en-US/DB6_Admin_Guide.pdf>

This excerpt below explains the logic and I feel this is a sound solution. Whether it meets all the requirements for regulations set by other entities can be discussed on a case by case basis.

“Db2 Audit and Audit Exceptions Background Prior to Db2 12.1, with Db2 audit enabled for the database using the db2audit facility, an unsustainably large amount of audit data is generated which has an impact on the general database performance, I/O, and space.   
  
This is especially true for the “execute” category of Db2 auditing. Almost all the data is generated by the SAP application and the SAP database connect user.   
  
Unfortunately, the db2audit facility provides very little in terms of application context that comes from the SAP system such as SAP logon user, reports, and so on. On the other hand, there are SAP-level auditing functions and auditing within the SAP system which can provide application-related details.   
  
However, auditing on SAP application level has its limits: You cannot audit statements that execute directly against the database and that come from outside the local ABAP stack. Enhancements as of Db2 12.1 With Db2 12.1, the Db2 audit facility has been enhanced to overcome the above-mentioned limitation. Generally, the database data should only be accessed in a trusted way from the SAP application through the SAP database connect user, and all other connections to the database should be audited.   
  
As of Db2 12.1, you can exclude auditing for connections originating from the local SAP application by the SAP connect user. This reduces the amount of audit data generated by the Db2 audit facility to something more manageable.”  
  
Process   
  
1. Define a trusted context using the trust procedure attribute SAPTOOLS.TRUST\_PROC.   
2. Establish a trusted connection.   
3. Create an audit exception for the trusted context.   
4. Define and activate the Db2 audit policy. For more information, see Creating Trusted Context and Verifying Trusted Connections and Creating, Verifying and Activating Audit Exceptions in the document.

### Minimum Software Requirements

The following SAP minimum releases are required:

SAP Application Server ABAP (AS ABAP): 7.00 or higher

SAP Application Server Java (AS Java): 7.50 or higher

You must use SL Toolset SP 42 or higher with Db2 12.1

Lower SAP releases are not supported with Db2 12.1.

You must use the following minimum levels:

SAP NetWeaver 7.00 SP27

SAP NetWeaver 7.01 SP12

SAP NetWeaver 7.02 SP12

SAP NetWeaver 7.31 SP05

SAP NetWeaver 7.40 SP02

SAP NetWeaver 7.50 and higher: No minimum Support Packages are required.

1. **Only if you want to perform an SAP upgrade to SAP NetWeaver 7.40 SP08 or higher, you can upgrade to DB2 12.1 on lower SAP Basis release levels immediately before you begin the SAP upgrade. Read SAP Note 2146598 for more information.** [**https://me.sap.com/notes/2146598**](https://me.sap.com/notes/2146598)
2. To be able to monitor a Db2 12.1 database using the DBA Cockpit, you must implement SAP Note 1835822 <https://me.sap.com/notes/2146598> in addition to the minimum SAP Basis release levels listed above. To be able to use the Monitoring and Alerting Infrastructure (MAI) of SAP Solution Manager 7.1 with Db2 12.1, you must apply SP9 or higher to SAP Solution Manager.
3. Make sure that you meet at least the minimum SAP levels listed in the prerequisites section. To take full advantage of the latest monitoring counters, you can upgrade to the support package levels listed in SAP Note 2827136.
4. Database upgrades to Db2 12.1 are possible from the following Db2 releases:

Db2 version 11.1

Db2 version 11.5

Follow the instructions in the database upgrade guide "Upgrading to Version 12.1 of IBM Db2 for Linux, UNIX, and Windows" that is available on SAP Help Portal at <https://help.sap.com/viewer/db6_upgrade_12_1>

See also the following important information in the IBM Db2 Documentation:

Db2 12.1 system requirements: See System requirements for IBM Db2 for Linux, UNIX, and Windows on the IBM Support website. <https://www.ibm.com/docs/en/db2>

Upgrade to Db2 12.1 <https://www.ibm.com/docs/en/db2/12.1?topic=fundamentals-upgrade-db2-version-121> in the IBM Db2 documentation <https://www.ibm.com/docs/en/db2>

Adjust your Db2 parameters according to SAP Note 3518384 <https://me.sap.com/notes/3518384>

when using DB2 12.1.   
  
You must enable Db2 Automatic Reorg to automate important index and table maintenance tasks such as type-2 index cleanup or space reclamation. For more information, see SAP Note 975352. <https://me.sap.com/notes/975352>

For best monitoring support in the DBA Cockpit (including metrics newly introduced with Db2 12.1), apply SAP Note 3532234 <https://me.sap.com/notes/3532234>

If you want to use the Db2 pureScale Feature, read the following information:

SAP guide for Db2 pureScale: <https://help.sap.com/viewer/p/DB6>

If you want to use SAP Business Warehouse (SAP BW) or applications based on SAP BW (SAP SCM, SAP SRM) with Db2 pureScale, see SAP Note 1716996 <https://me.sap.com/notes/1716996>

As of Db2 11.1.4.5, IBM Spectrum version 5.x will be bundled with Db2. IBM Spectrum version 5.x does not support RHEL 6.x and SLES 11.x. If you use these operating system levels, you must upgrade your operating system prior to the update to Db2 11.1.4.5. Read the following article for more information:

<https://www.ibm.com/support/pages/node/1073670?myns=swgimgmt&mynp=OCSSEPGG&mync=E&cm_sp=swgimgmt-_-OCSSEPGG-_-E>

The following SAP Notes contain other important information related to the use of IBM Db2 with SAP applications:

1555903 - Supported Db2 Database Features

101809 - Supported Db2 Versions and Fix Pack Levels

1168456 - Support Process and End of Support Dates for IBM Db2

1260217 - Software Components Contained in Db2 License from SAP

## Important note before starting.

This assumes the underlying filesystems for auditing (/db2/<SID>/AUDIT) exist. Do not use the default values supplied by IBM for the audit paths. Those point to the db2 filesystems used for the database. The below are the ones we will be using.

jq03a010:db2cai 17> db2audit describe

DB2 AUDIT SETTINGS:

Audit active: "TRUE "

Log audit events: "FAILURE"

[...]

Audit Data Path: "/db2/CAI/AUDIT/"

Audit Archive Path: "/db2/CAI/AUDIT/audarchive/"

How this works.   
  
With the trusted context set up in DB2 12.1 the changes made to the system following are written to the audit logs for both the database and the instance.

What is excluded:

1. changes made by a SAP work process.

What is included:

1. All changes made by the schema user (SAPCAP for CAI and CAP systems) at the operating systems level. This includes changes made by the [SYSTOOLS](#_93u61q7x4bmi) procedures and the import via tp or R3trans.
2. Also there’s a second policy in effect that is role-based which will slightly affect the volume of output.

How the audit records are handled

Audit records are written out to operating system files at a configured location. There are files for the instance and database. Further details are given later with examples.

We are using /db2/<SID>/AUDIT

EG: /db2/CAI/AUDIT for CAI.

This must be an independent filesystem from the database instance or the database files or SAP system binaries and working area, (/usr/sap/<SID> or /sapmnt/<SID>)

1. Every once in a while you will need to “archive” the **active audit files** for the instance and database files to a configured location on that (or another) filesystem. We have configured a subdirectory /db2/<SID>/AUDIT/audarchive in the policies to do this. Ideally this would be an altogether separate filesystem but that wasn’t done.
2. You need to flush the buffer and then archive the files to the “audarchive” directory. These are your single source of truth. These should never be deleted or must exist someplace for a successful audit. These will be referred to from now on as the **archived audit files**. All it is basically doing is moving the active file(s) to the “audarchive” directory and reinitializing it in the active audit file directory the next time it needs to write an entry. That may or may not immediately happen.
3. There are storage implications. You will need at least double the space of the archived log files if you have the “execute with data” setting activated.
4. We use the “audarchive” directory to extract the archived audit files to create a series of comma-delimited files for each of the audit categories which can then be extracted to eight different files which correspond to each of the audit tables. There’s an additional file “auditlobs” (no “del” extension) which contains the data captured when the “execute with data” or “context” option is chosen for a policy. This is appended to each of the files. So they can grow very, very large.   
     
   Some examples of the files are

validate.del

secmaint.del

objmaint.del

audit.del

sysadmin.del

checking.del

context.del

execute.del

auditlobs

The “execute.del” and “auditlobs” files can be very large and in some cases larger than the archived audit files as they aren’t compressed like the archive files are. The auditlobs file is a special case, it’s holding detailed data used for the context.del and execute.del. The latter two files have pointers to the file in “auditlobs” for that data. When it’s loaded into tables, the auditlobs data is added to the tables for the corresponding records.

1. These can be loaded into the corresponding database tables.

While it’s optional, using the archived audit files as a single source of truth will allow you to remove the \*.del files and truncate the database tables for use on a new set of queries. If this approach is chosen, you can always recreate the files later. Remember the \*.del files as well as the auditlobs are appended upon each extract. If repeatedly extracted files are loaded, you will almost certainly have duplicate records in the tables when loaded as there are no indexes or constraints on the audit tables.

1. There’s a requirement to send over the archived audit files on the production system to another server in India. I don’t have specifics on this as yet but it likely involves sending the files to the remote server and possibly cleaning up the archived audit files and any extracts on the in house auditing. At this point I have no idea what is in process.

Let’s delve into the mechanics of how this all works at a practical level next.

##### Setting up policies for India regulations:

I’m creating this policy strictly for the intended schema user but we aren’t auditing the schema user (SAPCAP) itself. In this case we are only monitoring the 52 tables requested. Do note the policy name matches the actual schema. At the time of this writing the specifications are for changes to 52 tables by the SAPCAP schema. But capturing the schema changes for the schema user results in a lot of “noise”, basically data that’s not required.

From a command line as the “db2<sid> user or one with SECADM and DBADM authorities. It should not be the schema user which should probably not have those authorizations in the first place.

db2 "CREATE AUDIT POLICY CAPSCHEMA \

CATEGORIES AUDIT STATUS NONE, \

CHECKING STATUS NONE, \

CONTEXT STATUS NONE, \

EXECUTE with data STATUS BOTH, \

SECMAINT STATUS NONE, \

SYSADMIN STATUS NONE \

ERROR TYPE AUDIT"

Also note that for this particular case we are only specifying “Execute with data”. “CONTEXT” is the old version of “Execute with data” and is superfluous. This can be changed on the fly to activate and deactivate each of the categories. You may still get entries for “context” but that may stem from what other audit policies or policies are active or from older archived audit files.

Next we have to add all the tables required.

You will run into various issues. The ones I have run into are:  
  
1. Auditors asking for tables that are actually part of a clustered table.   
 Fix: In this case, you have to specify the **entire** underlying cluster table.

2. Virtualized tables.  
 These will have zero records and are converted to views.   
 Fix: “Materialize” the table. (DB02 Tcode)

3. Materialized views.  
 Once again, audit the underlying tables.

Here’s a sample of adding a table.  
  
db2 "AUDIT TABLE SAPCAP.ANLA USING POLICY CAPSCHEMA"

The rest of the tables with notes on the clustered tables and a materialized view are available here:   
  
<https://docs.google.com/spreadsheets/d/1kf7fiFeFBjKYIeWM7656SNsGxoHSVFCbOpAa9v7tKCk/edit?gid=0#gid=0>

We don’t want to explicitly add the user. It’s not needed to just track individual tables. I’ll show you how to check this later once the tables are added. The commands below are here for your convenience.

From the db2 CLP prompt

To add:

audit user sapcap add policy

To remove:

audit user sapcap remove policy

DB2 Auditing - Logging Tables

The original request for tables is below

<https://docs.google.com/spreadsheets/d/1U1yylFgH2NMzlLKEfdqQXE9ShUzUtEeOXD8Y0KpdunQ/edit?gid=0#gid=0>

Note: Create the audit policy with an ID that has the SECADM authority (usually the db2<sid> user), the SAP schema user normally doesn’t have this. So in this case use the db2<sid> user.

I’m not creating this policy for the intended schema user, in this case SAPCAP, but for the tables if they are accessed by users outside of the SAP work processes. Do note the schema name matches the actual schema for the purposes of a consistent naming convention. We aren’t auditing the SAPCAP schema user, just the tables requested due to the ridiculous volume. Auditing at the OS level isn’t going to be as excessive and the auditors can get the entries from auditing (table logging flag in TCode SE13) in the SAP system instead for changes made by the SAP work processes.

db2 "CREATE AUDIT POLICY CAPSCHEMA \

CATEGORIES AUDIT STATUS NONE, \

CHECKING STATUS NONE, \

CONTEXT STATUS NONE, \

EXECUTE with data STATUS BOTH, \

SECMAINT STATUS NONE, \

SYSADMIN STATUS NONE \

ERROR TYPE AUDIT"

Auditing the usage of the schema user by DB2 auditing on an SAP system is asking for a lot of storage that will have to be transferred, parsed and reported.

So do not use this command for the schema user in this specific case. It’s included here for documentation purposes only.   
  
~~db2 "AUDIT USER SAPCAD USING POLICY SCHEMAPPOLICY"~~

We then have to enable auditing for the specific tables.

db2 "AUDIT TABLE SAPCAD.BSIS USING POLICY CAPSCHEMA"

DB20000I The SQL command completed successfully.

You may have a lot, in this case it’s 52 tables.

<https://docs.google.com/spreadsheets/d/1kf7fiFeFBjKYIeWM7656SNsGxoHSVFCbOpAa9v7tKCk/edit?gid=0#gid=0>

### Activate Auditing

##### Important note about the tools used in this document.

You will see usage in this document starting with the command line (Unix) tool “db2audit” and audit commands performed in the DB2 CLP (audit table XXXX)

The db2audit command is sometimes a wrapper for the CLP commands but not always. Some commands can only be done in one environment.   
  
For the command db2audit simply typing the command name with no parameters will get you the syntax of your commands.   
  
q03a010:db2cai 62> db2audit

AUD0002N Syntax error. Usage:

|>- db2audit -+- archive ---( Audit Archive )--------------------+-><|

+- configure -+- reset -----------------+----------+

| '-( Audit Configuration )-' |

+- describe ---------------------------------------+

+- extract -( Audit Extraction )-------------------+

+- flush ------------------------------------------+

+- start ------------------------------------------+

'- stop -------------------------------------------'

Audit Archive:

>--+----------------------------+-+------------------------------------+->

'- database -<database name>-' '- member -+-------------------------+

'-<current member number>-'

>--+-----------------------------+----------------------------------->

'- to --<audit-archive-path>--'

Audit Configuration:

>--------+------------------------------------------------------+->

| .-----,------------------------------------. |

| V | |

+- scope ---+-- all -------+-- status -+- both ----+-+-+

+-- audit -----+ +- none ----+

+-- checking --+ +- failure -+

+-- context --+ '- success -'

+-- objmaint --+

+-- secmaint --+

+-- sysadmin --+

'-- validate --'

>--+---------------------------+---------------------------------->

'- errortype -+- audit --+--'

'- normal -'

>--+--------------------------------+----------------------------->

'- datapath--<audit-data-path>---'

>--+----------------------------------------+--------------------->

'- archivepath--<audit-archive-path>-----'

Audit Extraction:

>--+----------------------------------------------------------------------+-->

+-file--<output file>--------------------------------------------------+

+-delasc---+-----------------------------+---+--------------------+----+

| '--delimiter-<load delimiter>-' '--to-<delasc path>--' |

'-syslog-<facility.priority>-+------------+-+------------------------+-'

'-tag-<word>-' '-splitrecordsize-<byte>-'

>--+---------------------------------------------------+---------->

+-status--+-success-+-------------------------------+

| '-failure-' |

| |

| .--,---------------------------------. |

| V | |

'-category--+-audit----+--+--------------------+-+--'

+-checking-+ '-status-+-failure-+-'

+-context--+ '-success-'

+-execute--+

+-objmaint-+

+-secmaint-+

+-sysadmin-+

'-validate-'

>--from--+------------------------+--files--<input log files>-----|

'--path-<archive path>---'

The DB2 CLP command “audit” is a limited subset of the db2audit command at OS level. It’s also not documented. There are also some operations you can only perform in the CLP.   
  
Frankly, the audit facility is clumsy and out of date as well as badly documented.

As a result, the “db2audit” command and the DB2 CLP command examples below are not always interchangeable.

Let’s move on to some of the practical uses.

##### For the instance:

At operating system level for the instance as the db2<sid> user.

db2audit start

##### For the database:

You can activate/enable auditing using the Db2 audit facility and your defined audit policy by running the following command to create the audit policy:

db2 "CREATE AUDIT POLICY CAPSCHEMA \

CATEGORIES AUDIT STATUS NONE, \

CHECKING STATUS NONE, \

CONTEXT STATUS NONE, \

EXECUTE with data STATUS BOTH, \

SECMAINT STATUS NONE, \

SYSADMIN STATUS NONE \

ERROR TYPE AUDIT"

And now implement the policy:

db2 “audit database using policy <your audit policy>”

jq03a010:db2cai 63> db2 "audit database using policy CAPSCHEMA"

DB20000I The SQL command completed successfully.

jq03a010:db2cai 64> db2 commit

DB20000I The SQL command completed successfully.

#### Stopping Auditing

##### For the instance

jq03a010:db2cai 65> db2audit stop

##### For the database

This is very involved. It requires you to remove the policy or policies you want removed. But there’s a hierarchy to delete the policy. Any tables, users or other objects attached to the policy have to be removed first. But once it’s active it stays active.

It is quite involved to remove..   
  
In the case of CAI or CAP, you have to drop all the table objects and any other objects.

There is an example provided on how to determine the objects attached to the policy.   
  
We will go through that quickly here.

jq03a010:db2cai 36> db2 ‘SELECT substr(AUDITPOLICYNAME,1,12) as "Policy ", \

OBJECTTYPE as "Obj Type", \

SUBOBJECTTYPE as "Sub Object Type", \

substr(OBJECTSCHEMA,1, 10) as "Schema", \

substr(OBJECTNAME,1,20) as "Object Name" \

FROM SYSCAT.AUDITUSE’

Horrible formatting below, it’s the best I can do with a word processor.

Policy Obj Type Sub Object Schema Object Name  
 Type

---------------- -------- --------------- ---------- -----------

ADMINSPOLICY i R - DB2\_SYSADM

CAPSCHEMA - CURRENT SERVER

CAPSCHEMA T SAPCAP BSAD

CAPSCHEMA T SAPCAP COBK

CAPSCHEMA T SAPCAP DBTABLOG

CAPSCHEMA T SAPCAP RFBLG

CAPSCHEMA T SAPCAP MSEG

[...]

CAPSCHEMA T SAPCAP DD02L

CAPSCHEMA T SAPCAP DD09L

CAPSCHEMA T SAPCAP DD04L

54 record(s) selected.

So to remove the auditing from the policy CAPSCHEMA, you would have to remove the user (if it’s coded) and all the tables.

### Dropping and removing Objects from an Audit Policy

Once you see what objects you have attached to the policy, it’s the same as adding only backwards.

#### For tables:

This is pretty straightforward from the db2 prompt as the db2cai (db2<sid>) user

db2 => audit table sapcap.bsad remove policy

Rinse, lather and repeat for the remaining tables.

#### For Users - Removing the database user from the database policy

One doesn’t exist in this case. **But let’s pretend one was added by mistake.**

jq03a010:db2cai 35> db2 "AUDIT USER SAPCAP USING POLICY CAPSCHEMA"

OOPS!

To remove the user from the policy:   
jq03a010:db2cai 30> db2 audit user sapcap remove policy

DB20000I The SQL command completed successfully.

#### Removing the database policy Let’s remove the **database policy** next. db2 => AUDIT DATABASE REMOVE POLICY <- Please read below

### Dropping the audit policy db2 drop audit policy CAPSCHEMA

jq03a010:db2cai 7> db2 drop audit policy CAPSCHEMA

DB20000I The SQL command completed successfully.

jq03a010:db2cai 8> db2 commit

DB20000I The SQL command completed successfully.

I always make sure a commit statement is issued in this case.

You need to stop and start the database instance to take effect.

#### Further analysis

This one is pretty misleading and of course isn’t documented. I managed to find someone who actually knew this and could get me squared away. You cannot purge the policy unless you purge this object:

We query syscat.audituse:

db2 ‘SELECT substr(AUDITPOLICYNAME,1,12) as "Policy ", \

OBJECTTYPE as "Obj Type", \

SUBOBJECTTYPE as "Sub Object Type", \

substr(OBJECTSCHEMA,1, 10) as "Schema", \

substr(OBJECTNAME,1,20) as "Object Name" \

FROM SYSCAT.AUDITUSE’  
  
  
Policy Obj Type Sub Object Schema Object Name  
 Type

---------------- -------- --------------- ---------- -----------

CAPSCHEMA - CURRENT SERVER

So the command(s) below is not exactly intuitive. As the db2<sid> user - first be sure you are connected as the db2<sid>.

jq03a010:db2cai 668> db2 set schema SAPCAP

DB20000I The SQL command completed successfully.

Let’s check!

jq03a010:db2cai 19> db2 values current user

1

---------

DB2CAI

1 record(s) selected.

jq03a010:db2cai 20> db2 values current schema

1

-------

SAPCAP

1 record(s) selected.

db2 => connect to cai

Database Connection Information

Database server = DB2/AIX64 12.1.0.0

SQL authorization ID = DB2CAI

Local database alias = CAI

db2 => AUDIT DATABASE REMOVE POLICY

**I did say it was confusing.**

##### Removing an Audit Exception for Trusted Context

You can remove the audit exception by running the following statement:

db2 "audit remove exception for trusted context DB6CTX\_SAP<SID>"

AUDIT DATABASE REMOVE POLICY

audit table DB2CAP.BASIS\_ADMINS remove policy

The below command works for every table listed below

To remove:  
audit table sapcad.COBK remove policy

### Queries for auditing

Let’s start by identifying the exposed tables and views that are used.   
  
Note:

Objects in the SYSIBM schema are system tables,   
Objects in the SYSCAT schema are system catalog views

SYSCAT.AUDITUSE  
SYSCAT.AUDITPOLICIES  
SYSCAT.AUDITEXCEPTIONS

SYSIBM.SYSAUDITEXCEPTIONS  
SYSIBM.SYSAUDITPOLICIES  
SYSIBM.SYSAUDITUSE

We will use the views because they are a simplification of the SYSIBM tables.

#### SYSCAT.AUDITUSE

db2 ‘SELECT substr(AUDITPOLICYNAME,1,12) as "Policy ", \

OBJECTTYPE as "Obj Type", \

SUBOBJECTTYPE as "Sub Object Type", \

substr(OBJECTSCHEMA,1, 10) as "Schema", \

substr(OBJECTNAME,1,20) as "Object Name" \

FROM SYSCAT.AUDITUSE’

The formatting is horrible so a sample screen shot is shown to minimize confusion.

Policy Obj Type Sub Object Type Schema Object Name

---------------------- ------------ ----------------------- —------- -----------

ADMINSPOLICY i R - DB2\_SYSADM

CAPSCHEMA i U - SAPCAP

CAPSCHEMA T SAPCAP BSAD

CAPSCHEMA T SAPCAP COBK

CAPSCHEMA T SAPCAP DBTABLOG

CAPSCHEMA T SAPCAP RFBLG

CAPSCHEMA T SAPCAP MSEG

[...]

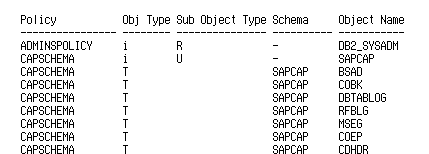
CAPSCHEMA T SAPCAP DD02L

CAPSCHEMA T SAPCAP DD09L

CAPSCHEMA T SAPCAP DD04L

Note the “role based” object DB2\_SYSADM for a different policy (ADMINSPOLICY).

54 record(s) selected.



Please note that we shouldn’t be tracking the user SAPCAP.   
We should remove this because it generates a ridiculous amount of logging that was never part of any specification.

##### SYSCAT.AUDITPOLICIES In this case, you will need to modify the AUDITPOLICYNAME variable to fit.

db2 “SELECT 'AUDITPOLICYNAME' AS "Policy ", \

CAST(AUDITPOLICYNAME AS VARCHAR(18)) AS "Value" \

FROM SYSCAT.AUDITPOLICIES WHERE AUDITPOLICYNAME = ‘CAPSCHEMA’ \

UNION all \

SELECT 'AUDITSTATUS' AS Policy, CAST(AUDITSTATUS AS VARCHAR(18)) AS VALUE \

FROM SYSCAT.AUDITPOLICIES WHERE AUDITPOLICYNAME = ‘CAPSCHEMA’ \

UNION all \

SELECT 'CONTEXTSTATUS' as Policy, CAST(CONTEXTSTATUS AS VARCHAR(18)) AS VALUE \

FROM SYSCAT.AUDITPOLICIES WHERE AUDITPOLICYNAME = ‘CAPSCHEMA’ \

UNION all \

SELECT 'VALIDATESTATUS' as Policy, CAST(VALIDATESTATUS AS VARCHAR(18)) AS VALUE \

FROM SYSCAT.AUDITPOLICIES WHERE AUDITPOLICYNAME = ‘CAPSCHEMA’ \

UNION all \

SELECT 'CHECKINGSTATUS' as Policy, CAST(CHECKINGSTATUS AS VARCHAR(18)) AS VALUE \

FROM SYSCAT.AUDITPOLICIES WHERE AUDITPOLICYNAME = ‘CAPSCHEMA’ \

UNION all \

SELECT 'SECMAINTSTATUS' as Policy, CAST(SECMAINTSTATUS AS VARCHAR(18)) AS VALUE \

FROM SYSCAT.AUDITPOLICIES WHERE AUDITPOLICYNAME = ‘CAPSCHEMA’ \

UNION all \

SELECT 'OBJMAINTSTATUS' as Policy, CAST(OBJMAINTSTATUS AS VARCHAR(18)) AS VALUE \

FROM SYSCAT.AUDITPOLICIES WHERE AUDITPOLICYNAME = ‘CAPSCHEMA’ \

UNION all \

SELECT 'SYSADMINSTATUS' as Policy, CAST(SYSADMINSTATUS AS VARCHAR(18)) AS VALUE \

FROM SYSCAT.AUDITPOLICIES WHERE AUDITPOLICYNAME = ‘CAPSCHEMA’ \

UNION all \

SELECT 'EXECUTESTATUS' as Policy, CAST(EXECUTESTATUS AS VARCHAR(18)) AS VALUE \

FROM SYSCAT.AUDITPOLICIES WHERE AUDITPOLICYNAME = ‘CAPSCHEMA’ \

UNION all \

SELECT 'EXECUTEWITHDATA' as Policy, CAST(EXECUTEWITHDATA AS VARCHAR(18)) AS VALUE \

FROM SYSCAT.AUDITPOLICIES WHERE AUDITPOLICYNAME = ‘CAPSCHEMA’ “

POLICY VALUE

---------------------------- ------------------

AUDITPOLICYNAME CAPSCHEMA

AUDITSTATUS N

CONTEXTSTATUS N

VALIDATESTATUS N

CHECKINGSTATUS N

SECMAINTSTATUS N

OBJMAINTSTATUS N

SYSADMINSTATUS N

EXECUTESTATUS B

EXECUTEWITHDATA Y

##### Please note that for this implementation we only include the “execute” settings. Dropping the audit policy db2 drop audit policy CAPSCHEMA

jq03a010:db2cai 7> db2 drop audit policy CAPSCHEMA

DB20000I The SQL command completed successfully.

jq03a010:db2cai 8> db2 commit

DB20000I The SQL command completed successfully.

I always make sure a commit statement is issued in this case.

##### SYSCAT.AUDITEXCEPTIONS

This tells you the name of the “trusted context”. It doesn’t tell you how the exception is implemented. Good luck finding it in their documentation.   
  
I found this in the db6\_update\_db\_out.log in the db2 home directory after executing the upgrade.   
  
CREATE TRUSTED CONTEXT DB6CTX\_SAPCAP BASED UPON CONNECTION USING SYSTEM AUTHID SAPCAP ENABLE ATTRIBUTES (TRUST PROCEDURE SAPTOOLS.TRUST\_PROC)

db2 'select substr(OBJECTNAME,1,8), substr(EXOBJECTNAME,1,20), substr(CREATE\_TIME,1,19) from syscat.AUDITEXCEPTIONS'  
  
Obj External Create

Name Obj. Name Time

-------- -------------------- --------------------

GLOBAL DB6CTX\_SAPCAP 2025-01-22-02.35.50

db2 "select substr(a.CONTEXTNAME,1,20) as CONTEXTNAME, substr(a.ATTR\_NAME,1,20) as ATTR\_NAME, substr(a.ATTR\_VALUE,1,30) as ATTR\_VALUE from syscat.contextattributes a where a.contextname <> 'SYSATSCONTEXT'"

CONTEXTNAME ATTR\_NAME ATTR\_VALUE

-------------------- -------------------- ------------------------------

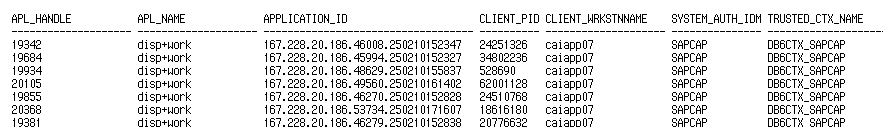
DB6CTX\_SAPCAP ENCRYPTION NONE

DB6CTX\_SAPCAP TRUST PROCEDURE SAPTOOLS.TRUST\_PROC

2 record(s) selected.

2 record(s) selected.

To verify a trusted connection between the SAP application server and the database using the database command line processor (CLP), run the following statement:

db2 "select substr(application\_handle,1,20) as apl\_handle, substr(application\_name,1,20) as apl\_name, substr(application\_id,1,35) as application\_id, substr(client\_pid,1,10) as client\_pid, substr(client\_wrkstnname, 1, 20) as client\_wrkstnname, substr(system\_auth\_id, 1, 8) as system\_auth\_idm, substr(trusted\_ctx\_name,1, 20) as trusted\_ctx\_name from table (mon\_get\_connection(null, null)) as t"  
  


## Appendix

### Testing scripts - not for production!

Thus far these only exist here: /db2/CAI/AUDIT on the caidb00.esc.win.colpal.com server.

1. truncateaudittables.sh - Cleans up audit tables prior to load and can also be used to clean up after an audit. This can also be used in a production setting.

To run: sh truncateaudittables.sh

##### Difference scripts

This is intended to be used only post upgrade of the DB. Using the db2audit.ddl I modified the script to create the audit tables as AUDIT.<TABLE\_NAME>\_NEW so you can use the scripts below to compare the current table column structures to the new ones.   
  
You can see that both sets of tables exist this way: db2 list tables for schema audit  
  
jq03a010:db2cai 10> db2 list tables for schema audit

Table/View Schema Type Creation time

------------------------------- --------------- ----- --------------------------

AUDIT AUDIT T 2024-05-15-22.48.40.060767

AUDIT\_NEW AUDIT T 2025-02-22-00.46.10.486647

CHECKING AUDIT T 2024-05-15-22.48.40.096089

CHECKING\_NEW AUDIT T 2025-02-22-00.46.10.568848

CONTEXT AUDIT T 2024-05-15-22.48.40.180862

CONTEXT\_NEW AUDIT T 2025-02-22-00.46.10.679859

EXECUTE AUDIT T 2024-05-15-22.48.40.208007

EXECUTE\_NEW AUDIT T 2025-02-22-00.46.10.708534

OBJMAINT AUDIT T 2024-05-15-22.48.40.111036

OBJMAINT\_NEW AUDIT T 2025-02-22-00.46.10.595530

SECMAINT AUDIT T 2024-05-15-22.48.40.128326

SECMAINT\_NEW AUDIT T 2025-02-22-00.46.10.617726

SYSADMIN AUDIT T 2024-05-15-22.48.40.147094

SYSADMIN\_NEW AUDIT T 2025-02-22-00.46.10.639594

VALIDATE AUDIT T 2024-05-15-22.48.40.162026

VALIDATE\_NEW AUDIT T 2025-02-22-00.46.10.659822

For each of the 8 tables involved there’s a sql script. You can combine them if you like. It assumes you have run the modified db2audit.ddl script and modified it to create the tables as <TABNAME>\_NEW in the AUDIT schema.  
auditdiff.sql

checkingdiff.sql

contextdiff.sql

executediff.sql

objmaintdiff.sql

secmaintdiff.sql

structdiff.sql

sysadmindiff.sql

What happens is that it compares the column names, types and lengths.

Generally this is only adding columns to the existing tables. In one case in one of the systems (not listed here but it was an 11.1.4 to 11.5.x upgrade) it changes a column from character to numeric.

With the code listing below, you can look at the table names highlighted in blue.

Sample:   
dbprd010:db2cap 46> db2 -tvf checkingdiff.sql; <- use “tf” if you don’t want verbose output

jq03a010:db2cai 22> pwd

/db2/CAI/AUDIT

jq03a010:db2cai 10> db2 -tvf auditdiff.sql

WITH audit\_table

AS

(

SELECT colno,

**Substr**(colname,1,20) AS columnname,

**Substr**(typename,1,20) AS TYPE,

length,

scale

FROM syscat.columns

WHERE tabschema = 'AUDIT'

AND tabname = 'AUDIT'

ORDER BY colno ), audit\_new\_table

AS

(

SELECT colno,

**Substr**(colname,1,20) AS columnname,

**Substr**(typename,1,20) AS TYPE,

length,

scale

FROM syscat.columns

WHERE tabschema = 'AUDIT'

AND tabname = 'AUDIT\_NEW'

ORDER BY colno )

SELECT **Coalesce**(s1.colno, s2.colno) AS colno,

**Coalesce**(s1.columnname, s2.columnname) AS columnname,

**Coalesce**(s1.TYPE, s2.TYPE) AS TYPE,

**Coalesce**(s1.length, s2.length) AS length,

**Coalesce**(s1.scale, s2.scale) AS scale,

CASE

WHEN s1.colno IS NULL THEN 'AUDIT\_NEW\_TABLE'

WHEN s2.colno IS NULL THEN 'AUDIT\_TABLE'

ELSE 'Both'

END AS difference\_source

FROM audit\_table s1

FULL OUTER JOIN audit\_new\_table s2

ON s1.colno = s2.colno

AND s1.columnname = s2.columnname

AND s1.TYPE = s2.TYPE

AND s1.le ngth = s2.length

AND s1.scale = s2.scale

WHERE s1.colno IS NULL

OR s2.colno IS NULL

OR s1.columnname <> s2.columnname

OR s1.TYPE <> s2.TYPE

OR s1.length <> s2.length

OR s1.scale <> s2.scale

ORDER BY 1,

2

COLNO COLUMNNAME TYPE LENGTH SCALE DIFFERENCE\_SOURCE

------ -------------------- -------------------- ----------- ------ ------------------

34 ACCESSCONTROLMGR INTEGER 4 0 CHECKING\_NEW\_TABLE

35 TENANTNAME VARCHAR 128 0 CHECKING\_NEW\_TABLE

2 record(s) selected.

In the rare case where a column is dropped in the original table the   
“DIFFERENCE\_SOURCE” would point to “CHECKING\_TABLE”

As can be seen the new audit.checking\_new table added 2 columns.

This is a quick and dirty solution and can always be improved.

Let’s fix the differences in the current tables! This is a sample from CAI. While you can also do the old drop and rename process, it’s just as easy to script these and run from a Unix command prompts as the db2<sid> .

db2 "alter table audit.audit add column tenantname varchar(128)"

db2 "alter table audit.checking add column ACCESSCONTROLMGR int"

db2 "alter table audit.checking add column tenantname varchar(128)"

db2 "alter table audit.context add column tenantname varchar(128)"

db2 "alter table audit.execute add column tenantname varchar(128)"

db2 "alter table audit.objmaint add column tenantname varchar(128)"

db2 "alter table audit.secmaint add column tenantname varchar(128)"

db2 "alter table audit.sysadmin add column tenantname varchar(128)"

db2 "alter table audit.validate add column tenantname varchar(128)"

You can rerun the scripts to verify.

The files in point #3 reside here: /db2/CAI/AUDIT/audarchive

##### purgeauditarch.sh and nulldelfiles.sh - Cleanup archived audit logs and extract files

Run as the db2<sid> or root. The purgeauditarch.sh script simply cleans up the audit archive logs (test only, not for production!).

The nulldelfiles.sh script nulls out the extract files. The nulldelfiles.sh script

can be used on production but be sure you understand the implications. As I mentioned before, the extract process appends to the existing \*.del files. So if someone audits 2015 to 2020 before you and you only want 2022, you might want to null them out if you don’t want to wade through hundreds of thousands of extraneous lines.   
  
Remember the single source of truth is still the archived log files.

### DB2 Upgrade to Version 12.1

<https://help.sap.com/doc/769ab46cf5ff405084e5d3a821705e52/12_1/en-US/DB6_Upgrade_12_1.pdf>

1555903

BC-DB-DB6

DB6: Supported IBM DB2 Database Features

1260217

BC-DB-DB6

DB6: Software Components Contained in DB2 License from SAP

1168456

BC-DB-DB6

DB6: Support Process and End of Support Dates for IBM DB2 LUW

101809

BC-DB-DB6

DB6: Supported DB2 Versions and Fix Pack Levels

This is an upgrade done in house for DB2 11.5, it’s pretty much the same. I have no idea where the actual upgrade document was placed.

<https://docs.google.com/document/d/1GbT3S1X0SLMN_yEY_QdxmeVG7woQ-ll4fxyJeSduXc8/edit?usp=drive_link>

You may want to look at this next link, it may or may not help. It’s basically screen shots.

<https://docs.google.com/document/d/1ALKEjYnNhjMxUDFvNIdMcB4Hdjw8XG6hYOtujiV4Cmw/edit?usp=drive_link>

## Steps for Auditing

There's two sets of auditing files at the OS Level.

Please note that in this specific case (CAI and CAP) much of this is in place.

There's two sets of auditing files at the OS Level

1. Instance level auditing

2. Database level (policies)

Let’s set up the auditing for Instance Level

##### 1. Instance level auditing

Set up AUDIT parameters at the instance level.

Here's the default settings for the instance:

db2audit describe

DB2 AUDIT SETTINGS:

Audit active: "**FALSE** "

Log audit events: "FAILURE"

Log checking events: "FAILURE"

Log object maintenance events: "FAILURE"

Log security maintenance events: "FAILURE"

Log system administrator events: "FAILURE"

Log validate events: "FAILURE"

Log context events: "NONE"

Return SQLCA on audit error: "FALSE "

Audit Data Path: ""

Audit Archive Path: ""

Current standard settings at Colgate - instance level

db2audit describe

DB2 AUDIT SETTINGS:

Audit active: "TRUE "

Log audit events: "BOTH"

Log checking events: "FAILURE"

Log object maintenance events: "FAILURE"

Log security maintenance events: "FAILURE"

Log system administrator events: "BOTH"

Log validate events: "FAILURE"

Log context events: "NONE"

Return SQLCA on audit error: "TRUE "

Audit Data Path: "/db2/CAP/AUDIT/"

Audit Archive Path: "/db2/CAP/AUDIT/audarchive/"

Note: for the “Audit active” at the instance level the commands are   
**db2audit stop** <- This will change the “Audit active: "TRUE "” to Audit active: "FALSE"  
**db2audit start** <- This will change the “Audit active: "FALSE "” to Audit active: "TRUE"

These need to be done with SECADM and/or DBADM rights. Don’t use the schema user.

At the instance level, the audit facility must be stopped and started explicitly by use of the **db2audit start** and **db2audit stop** commands. When you start instance-level auditing, the audit facility uses existing audit configuration information. Since the audit facility is independent of the Db2 database server, it will remain active even if the instance is stopped. In fact, when the instance is stopped, an audit record may be generated in the audit log.   
  
To start auditing at the database level, first you need to create an audit policy, then you associate this audit policy with the objects you want to monitor, such as, authorization IDs, database authorities, trusted contexts or particular tables. We are only concerned about the tables outlined in the somewhat defective original specifications.  
  
  
jq03a010:db2cai 42> db2audit stop

AUD0000I Operation succeeded.

jq03a010:db2cai 43> db2audit describe

DB2 AUDIT SETTINGS:

Audit active: "FALSE "

[...]

And:  
  
jq03a010:db2cai 45> db2audit describe

DB2 AUDIT SETTINGS:

Audit active: "TRUE "

[...]

We will need to change these by command settings.

Audit (AUDIT). Generates records when audit settings are changed or when the audit log is accessed.

Authorization Checking (CHECKING). Generates records during authorization checking of attempts to access or manipulate Db2 database objects or functions.

Object Maintenance (OBJMAINT). Generates records when creating or dropping data objects, and when altering certain objects.

Security Maintenance (SECMAINT). Generates records when:

Granting or revoking object privileges or database authorities

Granting or revoking security labels or exemptions

Altering the group authorization, role authorization, or override or restrict attributes of an LBAC security policy

Granting or revoking the SETSESSIONUSER privilege

Modifying any of the SYSADM\_GROUP, SYSCTRL\_GROUP, SYSMAINT\_GROUP, or SYSMON\_GROUP configuration parameters.

System Administration (SYSADMIN). Generates records when operations requiring SYSADM, SYSMAINT, or SYSCTRL authority are performed.

User Validation (VALIDATE). Generates records when authenticating users or retrieving system security information.

Operation Context (CONTEXT). Generates records to show the operation context when a database operation is performed. This category allows for better interpretation of the audit log file. When used with the log's event correlator field, a group of events can be associated back to a single database operation. For example, a query statement for dynamic queries, a package identifier for static queries, or an indicator of the type of operation being performed, such as CONNECT, can provide needed context when analyzing audit results.

For Status there are 4 possible values:

both

none

failure

success

For errortype (we use "audit")

audit

normal

db2audit configure scope audit status both errortype audit

AUD0000I Operation succeeded.

db2audit configure archivepath '/db2/CAD/AUDIT/audarchive'

AUDIT PATH:

db2audit configure datapath <AuditPath> (EG: /db2/<SID>/AUDIT )

DATA PATH:

db2audit configure archivepath <AuditPath> (EG: /db2/<SID>/AUDIT/audaudit )

Practical example as the db2<sid> user

db2audit configure auditpath /db2/$DB2DBDFT/AUDIT

Others are

Audit (AUDIT). Generates records when audit settings are changed or when the audit log is accessed.

Authorization Checking (CHECKING). Generates records during authorization checking of attempts to access or manipulate Db2 database objects or functions.

Object Maintenance (OBJMAINT). Generates records when creating or dropping data objects, and when altering certain objects.

Security Maintenance (SECMAINT). Generates records when:

Granting or revoking object privileges or database authorities

Granting or revoking security labels or exemptions

Altering the group authorization, role authorization, or override or restrict attributes of an LBAC security policy

Granting or revoking the SETSESSIONUSER privilege

Modifying any of the SYSADM\_GROUP, SYSCTRL\_GROUP, SYSMAINT\_GROUP, or SYSMON\_GROUP configuration parameters.

System Administration (SYSADMIN). Generates records when operations requiring SYSADM, SYSMAINT, or SYSCTRL authority are performed.

User Validation (VALIDATE). Generates records when authenticating users or retrieving system security information.

Operation Context (CONTEXT). Generates records to show the operation context when a database operation is performed. This category allows for better interpretation of the audit log file. When used with the log's event correlator field, a group of events can be associated back to a single database operation. For example, a query statement for dynamic queries, a package identifier for static queries, or an indicator of the type of operation being performed, such as CONNECT, can provide needed context when analyzing audit results.

##### 2. Database level (policies)

## Testing that auditing works.

Given that we are auditing only the selected tables, a simple test of updating a single field and backing out the change should suffice.

I picked a record from CDPOS from client 000 as the likely candidate with a simple change to the ID.   
  
Modify as needed. The original record had the ID APSHF00 and we’re merely going to change it in a minor fashion and reverse it out. We want to do this as the schema user in this case.

db2 "update sapcap.cdhdr set username = 'APSHF001' where mandant = '000' and objectclas = 'ADRESSE3' and objectid = 'BC0100000222650000009921' and username = 'APSHF00'"

db2 commit

db2 "update sapcap.cdhdr set username = 'APSHF00' where mandant = '000' and objectclas = 'ADRESSE3' and objectid = 'BC0100000222650000009921' and username = 'APSHF001'"

db2 commit

Next we flush the buffer

db2audit flush

AUD0000I Operation succeeded.

Next let’s look at the active log file - note that it records the statement as entered, it does not change the case of the statement which complicates the search once loaded to tables. We will need to use an “UPPER” statement to extract the data. It shouldn’t matter from a performance standpoint, the tables have no indexes. You’re scanning the whole table anyway.

strings db2audit.db.CAI.log.0 | grep APSHF

¢update sapcap.cdhdr set username = 'APSHF001' where mandant = '000' and objectclas = 'ADRESSE3' and objectid = 'BC0100000222650000009921' and username = 'APSHF00'

£update sapcap.cdhdr set username = 'APSHF00' where mandant = '000' and objectclas = 'ADRESSE3' and objectid = 'BC0100000222650000009921' and username = 'APSHF001'

¢update sapcap.cdhdr set username = 'APSHF001' where mandant = '000' and objectclas = 'ADRESSE3' and objectid = 'BC0100000222650000009921' and username = 'APSHF00'

£update sapcap.cdhdr set username = 'APSHF00' where mandant = '000' and objectclas = 'ADRESSE3' and objectid = 'BC0100000222650000009921' and username = 'APSHF001'

We can archive it and extract to comma-delimited files (EG: execute.del and auditlobs plus the other 8 tables.   
  
Let’s try it just to get a taste of the process.

It is important to remember that the \*.del files the archived logs are extracted to are appended to each other. As I consider the \*.del files readable but mostly intended for loading to database tables, you can opt to recreate them each time or even archive them with timestamps. Be aware the huge size of the files does not lend itself to loading a comma-delimited file into a spreadsheet if the spreadsheet software has a hard limit for a number of rows.

But the likely dates for the data desired will likely slide as the years go on and the volume of the audit logs are so large that you will likely want to reinitialize the \*.del extract files and the contents of the tables. Remember that you can rebuild the \*.del files and the tables as long as you have the archived audit logs.

Let’s review the commands to archive this:

1. db2audit flush <- Flush the audit buffers

2. db2audit archive <- INSTANCE LEVEL LOGS

3. db2audit archive database <SID> <- DATABASE LEVEL LOGS

Let’s see it in action:

jq03a010:db2cai 67> db2audit flush

AUD0000I Operation succeeded.

jq03a010:db2cai 68> db2audit archive

Member DB Partition AUD Archived or Interim Log File

Number Number Message

-------- -------------- -------- -------------------------------------------------

0 0 AUD0000I db2audit.instance.log.0.20250213032936

AUD0000I Operation succeeded.

jq03a010:db2cai 69> db2audit archive database CAI

Member DB Partition AUD Archived or Interim Log File

Number Number Message

-------- -------------- -------- -------------------------------------------------

0 0 AUD0000I db2audit.db.CAI.log.0.20250213032948

AUD0000I Operation succeeded.

jq03a010:db2cai 70> ls -ltra

After the operations are performed, you should not see the old instance or database log in the /db2/<SID>/AUDIT directory. There may be new ones that are very small if anything has been written out. **If the old ones remain there, it’s possible the filesystem is full.** (It’s basically moved to the configured audit archive directory of:  
  
/db2/<SID>/AUDIT/audarchive  
  
jq03a010:db2cai 73> cd audarchive

jq03a010:db2cai 74> ls -ltra

drwxr-xr-x 2 db2cai dbcaiadm 256 Feb 12 02:10 old

-rw------- 1 db2cai dbcaiadm 12548 Feb 12 19:55 db2audit.instance.log.0.20250212195542

-rw------- 1 db2cai dbcaiadm 8552 Feb 13 03:29 db2audit.instance.log.0.20250213032936

drwxr-xr-x 3 db2cai dbcaiadm 4096 Feb 13 03:29 .

-rw------- 1 db2cai dbcaiadm 947078462 Feb 13 03:29 db2audit.db.CAI.log.0.20250213032948

drwxr-xr-x 5 db2cai dbcaiadm 4096 Feb 13 03:31 ..

Also understand that archiving a file is actually a “move” statement in two parts. So it requires free space at least the size of the active log you are archiving.

Under the covers the above command (db2audit archive database <SID>) performs a copy of the file to the new destination with a timestamp added and only after checking the destination file if OK will remove the source file.

In pseudocode

db2audit archive database CAI performs the following:  
  
mv <active log file> <archive log file+timestamp>

check destination file

rm <active log file>

This is a drastically shortened version to just illustrate the point.

Let’s assume you have a 27GB filesystem and this includes the /db2/CAI/AUDIT and the /db2/CAI/AUDIT/audarchive directory.  
  
  
jq03a010:db2cai 2> ls -l

drwxr-xr-x 3 db2cai dbcaiadm 4096 Feb 14 20:12 audarchive

-rw------- 1 db2cai dbcaiadm 18062246344 Feb 18 10:55 db2audit.db.CAI.log.0

-rwxrwxrwx 1 db2cai dbcaiadm 17617 Jun 24 2024 db2audit.ddl

-rw------- 1 db2cai dbcaiadm 8372 Feb 14 20:06 db2audit.instance.log.0

Note we have an 18GB file. You really need 19GB free space in the destination directory of that filesystem to move the file to the destination directory. That it happens to be on the same filesystem in this case is irrelevant.

Let’s watch what happens when there isn’t enough space to complete the operation. We are in the active log directory and trying to move the file to the destination directory.   
  
Note we are moving the file in /db2/<SID>/AUDIT/ to /db2/<SID>/AUDIT/audarchive and are in the /db2/<SID>/AUDIT/ directory.

jq03a010:db2cai 3> db2audit archive database CAI

Member DB Partition AUD Archived or Interim Log File

Number Number Message

-------- -------------- -------- -------------------------------------------------

0 0 AUD0004N db2audit.db.CAI.log.0.20250218105626236609.bk

AUD0001N Operation failed.

Note that the file is created with a timestamp and an extension of “bk”. This is in the source directory of /db2/<SID>/AUDIT/ though.

jq03a010:db2cai 4> ls -ltra

-rw------- 1 db2cai dbcaiadm 18062246976 Feb 18 10:56 db2audit.db.CAI.log.0.20250218105626236609.bk

drwxr-xr-x 3 db2cai dbcaiadm 4096 Feb 18 10:57 audarchive

-rw------- 1 db2cai dbcaiadm 8584 Feb 18 10:57 db2audit.instance.log.0

jq03a010:db2cai 5>

So what happened is that the destination directory ran out of space and the file was renamed in the source directory. To fix this you can add more space and ensure your archiving destination has enough space. Then copy the file over without the \*.bk extension to the archive directory   
(/db2/<SID>/AUDIT/ audarchive in this case.) and move on.

Don’t ask how this all works, it’s not explained anywhere I could find.

### Extract Archived Logs to Readable Format

Note that this is for test purposes.

Let’s do the database files first, please note you can wildcard them. Also note that multiple database policies on the same database and the criteria for each will write to the same audit logging files.   
  
jq03a010:db2cai 75> db2audit extract delasc to /db2/CAI/AUDIT/audarchive from files /db2/CAI/AUDIT/audarchive/db2audit.db.CAI.log.0.\*

AUD0000I Operation succeeded.

jq03a010:db2cai 76> ls -ltra

drwxr-xr-x 2 db2cai dbcaiadm 256 Feb 12 02:10 old

-rw------- 1 db2cai dbcaiadm 12548 Feb 12 19:55 db2audit.instance.log.0.20250212195542

-rw------- 1 db2cai dbcaiadm 8552 Feb 13 03:29 db2audit.instance.log.0.20250213032936

-rw------- 1 db2cai dbcaiadm 947078462 Feb 13 03:29 db2audit.db.CAI.log.0.20250213032948

drwxr-xr-x 5 db2cai dbcaiadm 4096 Feb 13 03:31 ..

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 validate.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 secmaint.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 objmaint.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 checking.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 audit.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 sysadmin.del

drwxr-xr-x 3 db2cai dbcaiadm 4096 Feb 13 03:36 .

-rw-rw---- 1 db2cai dbcaiadm 72188 Feb 13 03:37 context.del

-rw-rw---- 1 db2cai dbcaiadm 758971270 Feb 13 03:37 auditlobs

-rw-rw---- 1 db2cai dbcaiadm 892600604 Feb 13 03:37 execute.del

We will proceed with the instance files next. Pay attention to the difference in the file sizes of the “\*.ddl” extract files after the instance files are added. You will need at least double the space of the archived log files if you have the “execute with data” setting activated.

jq03a010:db2cai 77> db2audit extract delasc to /db2/CAI/AUDIT/audarchive from files /db2/CAI/AUDIT/audarchive/db2audit.instance\*

AUD0000I Operation succeeded.

jq03a010:db2cai 78> ls -ltra

drwxr-xr-x 2 db2cai dbcaiadm 256 Feb 12 02:10 old

-rw------- 1 db2cai dbcaiadm 12548 Feb 12 19:55 db2audit.instance.log.0.20250212195542

-rw------- 1 db2cai dbcaiadm 8552 Feb 13 03:29 db2audit.instance.log.0.20250213032936

-rw------- 1 db2cai dbcaiadm 947078462 Feb 13 03:29 db2audit.db.CAI.log.0.20250213032948

drwxr-xr-x 5 db2cai dbcaiadm 4096 Feb 13 03:31 ..

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 secmaint.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 objmaint.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 checking.del

drwxr-xr-x 3 db2cai dbcaiadm 4096 Feb 13 03:36 .

-rw-rw---- 1 db2cai dbcaiadm 72188 Feb 13 03:37 context.del

-rw-rw---- 1 db2cai dbcaiadm 758971270 Feb 13 03:37 auditlobs

-rw-rw---- 1 db2cai dbcaiadm 892600604 Feb 13 03:37 execute.del

-rw-rw---- 1 db2cai dbcaiadm 570 Feb 13 03:39 validate.del

-rw-rw---- 1 db2cai dbcaiadm 352 Feb 13 03:39 audit.del

-rw-rw---- 1 db2cai dbcaiadm 7049 Feb 13 03:39 sysadmin.del

As shown above, the audit categories had a few entries written.

Before extraction, it’s advisable to make sure no tablespaces, particularly the one used for the audit schema, are not in any state but 0x0000 - I will explain in a moment why.

This is a quick and dirty command below to check.

Note: these are all AIX commands like below.

jq03a010:db2cai 27> db2 list tablespaces |grep -i state | grep -v 0x0000

[no output]

A lack of output is the desired result.

Let’s make sure your current user is the db2<sid>, you don’t want the actions reflected in the extracts and subsequent tables.

jq03a010:db2cai 28> db2 values current user

1

-------------

DB2CAI

1 record(s) selected.

Let’s set the schema to audit to avoid any embarrassing mistakes on a production box.

jq03a010:db2cai 29> db2 set current schema = 'AUDIT'

Let’s set the user to the proper schema just in case.

Check:

jq03a010:db2cai 30> db2 values current schema

1

-------------------------

AUDIT

1 record(s) selected.

Let’s list our tables:

jq03a010:db2cai 31> db2 list tables for schema audit

Table/View Schema Type Creation time

------------------------------- --------------- ----- --------------------------

AUDIT AUDIT T 2024-05-15-22.48.40.060767

CHECKING AUDIT T 2024-05-15-22.48.40.096089

CONTEXT AUDIT T 2024-05-15-22.48.40.180862

EXECUTE AUDIT T 2024-05-15-22.48.40.208007

OBJMAINT AUDIT T 2024-05-15-22.48.40.111036

SECMAINT AUDIT T 2024-05-15-22.48.40.128326

SYSADMIN AUDIT T 2024-05-15-22.48.40.147094

VALIDATE AUDIT T 2024-05-15-22.48.40.162026

9 record(s) selected.

This is for the purpose of testing and cleaning up the tables while no audit is in process by auditors, we can clean out the tables quickly.   
  
So let’s truncate the tables just to avoid confusion. Remember that the single source of truth are the archived auditlogs. These are from the command prompt and preferably not using the schema user although it may not make a difference.

db2 "truncate table audit.AUDIT drop storage immediate"

db2 "truncate table audit.CHECKING drop storage immediate"  
db2 "truncate table audit.CONTEXT drop storage immediate"  
db2 "truncate table audit.EXECUTE drop storage immediate"

db2 "truncate table audit.OBJMAINT drop storage immediate"

db2 "truncate table audit.SECMAINT drop storage immediate"  
db2 "truncate table audit.SYSADMIN drop storage immediate"  
db2 "truncate table audit.VALIDATE drop storage immediate"

Let’s load the files to audit tables now.

The first example below is the simplest. It creates a set of 8 comma delimited files. If “execute with data” is part of the audit policy the command above will also create a file called “auditlobs” which contains the values used.

From the command prompt as the db2<sid> user:

For the file: db2audit.db.CAI.log.0.20250214025357

jq03a010:db2cai 44> db2audit extract **delasc** to /db2/CAI/AUDIT/audarchive from files /db2/CAI/AUDIT/audarchive/db2audit.db.CAI.log.0.20250214025357

You can wildcard the db2audit.\* files and you’ll see that in the next example using a stored procedure.

Here’s an example of an extract for a year for all types of logs, be it instance or database.

CALL SYSPROC.AUDIT\_DELIM\_EXTRACT(

'', '', '/auditarchive', 'db2audit.%.200604%', '' )

In another example, they can call the SYSPROC.AUDIT\_DELIM\_EXTRACT stored procedure to extract the archived audit records with success events from the EXECUTE category and failure events from the CHECKING category, from a file with the timestamp they are interested in:

CALL SYSPROC.AUDIT\_DELIM\_EXTRACT( '', '', '/auditarchive',

'db2audit.%.20060419034937', 'category

execute status success, checking status failure );

Mind you, it’s a little more involved but will make for a slightly smaller output. Since we are concerned with the output of the SQL statements and thus the “execute.del” file, 99% of the output is this anyway. The “call sysproc” examples given above are there for anyone needing them in the future.

Let’s get back to the current operation and extract the audit file(s) to comma delimited files.

jq03a010:db2cai 13> ls -l

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 21:24 audit.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 21:19 auditlobs

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 checking.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 21:20 context.del

-rw------- 1 db2cai dbcaiadm 1406994182 Feb 14 02:54 db2audit.db.CAI.log.0.20250214025357

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 21:19 execute.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 objmaint.del

drwxr-xr-x 2 db2cai dbcaiadm 256 Feb 12 02:10 old

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 secmaint.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 21:20 sysadmin.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 21:20 validate.del

jq03a010:db2cai 14> db2audit extract delasc to /db2/CAI/AUDIT/audarchive from files /db2/CAI/AUDIT/audarchive/db2audit.db.CAI.log.0.20250214025357

AUD0000I Operation succeeded.

jq03a010:db2cai 15> ls -ltra

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 secmaint.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 objmaint.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 03:36 checking.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 21:20 sysadmin.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 21:20 validate.del

-rw-rw---- 1 db2cai dbcaiadm 0 Feb 13 21:24 audit.del

drwxr-xr-x 3 db2cai dbcaiadm 4096 Feb 14 02:53 .

-rw------- 1 db2cai dbcaiadm 1406994182 Feb 14 02:54 db2audit.db.CAI.log.0.20250214025357

drwxr-xr-x 5 db2cai dbcaiadm 4096 Feb 14 02:55 ..

-rw-rw---- 1 db2cai dbcaiadm 96248 Feb 14 02:57 context.del

-rw-rw---- 1 db2cai dbcaiadm 1323595385 Feb 14 02:57 execute.del

-rw-rw---- 1 db2cai dbcaiadm 1127742490 Feb 14 02:57 auditlobs

As seen above you can see that the last three files were populated but had the archived audit log from the instance had been included, the other zero byte file(s) would likely have data as well.

We will concentrate on loading the data into the audit.context and audit.execute tables. Those to comma delimited files have pointers to locations in the auditlobs file and when it’s loaded into the tables, the auditlobs file will also populate the tables.

Remember, we truncated the audit tables earlier.   
  
It won’t hurt to verify the tablespaces are all at “0”, someone may be taking a backup or something similar.   
  
You can check the tablespace states quickly by   
jq03a010:db2cai 666> db2 list tablespaces |grep -i state

Or you can look for anything not equal to State = 0x0000

Jq03a010:db2cai 667> db2 list tablespaces |grep -i state | grep -v 0x0000

You should have no output from the above command.

Remember that you want to be in the db2<sid> ID.   
  
jq03a010:db2cai 668> db2 set schema audit

DB20000I The SQL command completed successfully.

Let’s check!

jq03a010:db2cai 19> db2 values current user

1

---------

DB2CAI

1 record(s) selected.

jq03a010:db2cai 20> db2 values current schema

1

-------

AUDIT

1 record(s) selected.

Excellent!  


Let’s extract the files to the two tables. Pay attention to the syntax, particularly the keyword LOBSINFILE. That’s how you get the detailed data from the “auditlobs” file. The LOBSINFILE keyword only really applied to the “context.del” and “execute.del” files.

Please also note that there are no indexes on the tables. So you can get duplicate data in the tables. This is another reason to clean the tables out. Indexes may not work out depending on the data you are querying. If there are a lot of duplicate values, you are probably wasting time. If you are scanning on the LOB field, it’s probably going to a full scan anyway.

Very important is the keyword “nonrecoverable”. You must include that or you will put whatever tablespace the tables exist in into a LOAD [PENDING](#_6pvgtpavis9w) state which requires you to remove the state at some point by [various techniques](#_6pvgtpavis9w), As these are standalone tables it’s much simpler to add the clause.

db2 “LOAD FROM /db2/CAI/AUDIT/audarchive/context.del OF del MODIFIED BY LOBSINFILE INSERT INTO audit.context nonrecoverable”

db2 “LOAD FROM /db2/CAI/AUDIT/audarchive/execute.del OF del MODIFIED BY LOBSINFILE INSERT INTO audit.execute nonrecoverable”

jq03a010:db2cai 21> db2 "LOAD FROM /db2/CAI/AUDIT/audarchive/context.del OF del MODIFIED BY LOBSINFILE INSERT INTO audit.context nonrecoverable"

SQL3109N The utility is beginning to load data from file

"/db2/CAI/AUDIT/audarchive/context.del".

SQL3500W The utility is beginning the "LOAD" phase at time "02/14/2025

03:28:34.552668".

SQL3519W Begin Load Consistency Point. Input record count = "0".

SQL3520W Load Consistency Point was successful.

SQL3114W Some data following "","" in row "111" and column "1" was not

loaded.

SQL3114W Some data following "","" in row "181" and column "1" was not

loaded.

SQL3110N The utility has completed processing. "426" rows were read from the

input file.

SQL3519W Begin Load Consistency Point. Input record count = "426".

SQL3520W Load Consistency Point was successful.

SQL3515W The utility has finished the "LOAD" phase at time "02/14/2025

03:28:34.591292".

SQL3107W At least one warning message was encountered during LOAD processing.

Number of rows read = 426

Number of rows skipped = 0

Number of rows loaded = 426

Number of rows rejected = 0

Number of rows deleted = 0

Number of rows committed = 426

That’s relatively short, the next one is HUGE.

jq03a010:db2cai 22> db2 "LOAD FROM /db2/CAI/AUDIT/audarchive/execute.del OF del MODIFIED BY LOBSINFILE INSERT INTO audit.execute nonrecoverable"

SQL3109N The utility is beginning to load data from file

"/db2/CAI/AUDIT/audarchive/execute.del".

SQL3500W The utility is beginning the "LOAD" phase at time "02/14/2025

03:29:06.331999".

SQL3519W Begin Load Consistency Point. Input record count = "0".

SQL3520W Load Consistency Point was successful.

SQL3114W Some data following "","" in row "711" and column "1" was not

loaded.

SQL3125W The character data in row "711" and column "19" was truncated

because the data is longer than the target database column.

[...]

SQL3114W Some data following "","" in row "5389270" and column "1" was not

loaded.

SQL3125W The character data in row "5389270" and column "19" was truncated

because the data is longer than the target database column.

SQL3110N The utility has completed processing. "5389362" rows were read from

the input file.

SQL3519W Begin Load Consistency Point. Input record count = "5389362".

SQL3520W Load Consistency Point was successful.

SQL3515W The utility has finished the "LOAD" phase at time "02/14/2025

03:30:41.554573".

SQL3107W At least one warning message was encountered during LOAD processing.

Number of rows read = 5389362

Number of rows skipped = 0

Number of rows loaded = 5389362

Number of rows rejected = 0

Number of rows deleted = 0

Number of rows committed = 5389362

jq03a010:db2cai 23>

For the other 6 tables, the syntax goes something like this, you don’t need the “LOBSINFILE” clause. :  
  
db2 “LOAD FROM /db2/CAI/AUDIT/audarchive/checking.del OF del MODIFIED BY DELPRIORITYCHAR INSERT INTO audit.checking nonrecoverable”

### Query Loaded Tables.

This is going to vary depending upon what you are looking for but let’s take this specific case to address the SAPCAP schema and the execute table.

Now that we loaded these into tables. We can begin to query them.  
  
Because there’s a lot of “noise” in the auditing from the [SYSTOOLS](#_93u61q7x4bmi) schema, we want to limit the output to the SAPCAP schema in this case.

Here’s a sample:

db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10) AS USER, \

SUBSTR(AUTHID,1,10) as AUTHID, SUBSTR(STMTTEXT,1,110) AS SQL FROM AUDIT.EXECUTE \

where USERID = 'SAPCAP' or USERID = 'sapcap' \

and substr(STMTTEXT,1,6) in ('UPDATE','INSERT','DELETE')"

jq03a010:db2cai 23> db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10) AS USER, \

SUBSTR(AUTHID,1,10) as AUTHID, SUBSTR(STMTTEXT,1,110) AS SQL FROM AUDIT.EXECUTE \

where UPPER(substr(STMTTEXT,1,6)) in ('UPDATE','INSERT','DELETE') \

and STMTTEXT like '%APSHF00%'"

Note: the LIKE statement at the end assumes you know the ID of the person.

If you want to not display the SYSTOOLS in the query results, you can do this:

db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10) AS USER, \

SUBSTR(AUTHID,1,10) as AUTHID, SUBSTR(STMTTEXT,1,110) AS SQL FROM AUDIT.EXECUTE \

where UPPER(substr(STMTTEXT,1,6)) in ('UPDATE','INSERT','DELETE') \

and STMTTEXT **NOT like '%SYSTOOLS.%' or** STMTTEXT **NOT like '%SESSION%' or \**

STMTTEXT **NOT like '%SYSINDEXES%'**”

TIMESTAMP USER AUTHID SQL

-------------------------- ---

0 record(s) selected.

In this case, you may get nothing. It’s a test system. But the principle is the same for the context table.

db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10) AS USER, \

SUBSTR(AUTHID,1,10) as AUTHID, SUBSTR(STMTTEXT,1,110) AS SQL FROM AUDIT.CONTEXT \

where UPPER(substr(STMTTEXT,1,6)) in ('UPDATE','INSERT','DELETE') \

and STMTTEXT like '%APSHF00%'"

This is just an example to illustrate just how much there is data that is superfluous to what is desired from the db2 auditing.

## Troubleshooting, Assorted commands and Notes

### Assorted commands - Reference

Be aware that DB2 is on multiple platforms, the syntax is not consistent. Thus when searching on commands, it’s important to be sure it applies to the DB2 LUW platform. This especially applies to any AI queries or searches.

Be careful.   
  
  
I’m going to do this in an order of usage as if you are setting up a system. It assumes a distinct filesystem for the audit logs, both for active and archived logs, exists.

#### Create or adjust the AUDIT buffer in the DBM

jq03a010:db2cai 30> db2 "update dbm cfg using audit\_buf\_sz 1000"

DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed successfully.

You can check:   
  
jq03a010:db2cai 31> db2 get dbm cfg |grep -i audit\_buf\_sz

Audit buffer size (4KB) (AUDIT\_BUF\_SZ) = 1000

This requires a restart of the db2 instance.

Let’s start with the further setup for instance auditing

Set up AUDIT parameters.

jq03a010:db2cai 33> db2audit configure scope sysadmin status both error type audit

The AUD0000I Operation succeeded.

#### Configure the data path and the archive path.

jq03a010:db2cai 31> db2audit configure datapath /db2/CAI/AUDIT

The AUD0000I Operation succeeded.

jq03a010:db2cai 31> db2audit configure archivepath /db2/CAI/AUDIT/audarchive

#### The AUD0000I Operation succeeded. Start or stop instance level auditing.

jq03a010:db2cai 34> db2audit stop

The AUD0000I Operation succeeded.

jq03a010:db2cai 35> db2audit start

The AUD0000I Operation succeeded.

##### Show instance level auditing

jq03a010:db2cai 36> db2audit describe

DB2 AUDIT SETTINGS:

Audit active: "TRUE "

Log audit events: "FAILURE"

Log checking events: "FAILURE"

Log object maintenance events: "FAILURE"

Log security maintenance events: "FAILURE"

Log system administrator events: "FAILURE"

Log validate events: "FAILURE"

Log context events: "NONE"

Return SQLCA on audit error: "FALSE "

Audit Data Path: "/db2/CAI/AUDIT/"

Audit Archive Path: "/db2/CAI/AUDIT/audarchive/"

The AUD0000I Operation succeeded.

#### Create an audit policy for the database auditing.

#### 

db2 "CREATE AUDIT POLICY CAPSCHEMA \

CATEGORIES AUDIT STATUS NONE, \

CHECKING STATUS NONE, \

CONTEXT STATUS NONE, \

EXECUTE with data STATUS BOTH, \

SECMAINT STATUS NONE, \

SYSADMIN STATUS NONE \

ERROR TYPE AUDIT"

#### Flush audit buffers - (before archiving active logs)

jq03a010:db2cai 31> db2audit flush

The AUD0000I Operation succeeded.

This takes care of the buffer which means that both the instance and the database buffers are flushed. I strongly suggest this be done as well before archiving the active logs to get a complete snapshot at that timestamp.

#### Move Active Audit Files to Archive

jq03a010:db2cai 35> db2audit archive <- INSTANCE LEVEL LOGS

jq03a010:db2cai 36> db2audit archive database <SID> <- DATABASE LEVEL LOGS

Optionally you can archive to a different location by overriding the default audit archive location. This is a temporary alternative if you have a space issue. This example is contrived as a warning. *You should not archive to a location where the active database files or binaries are located.*  
jq03a010:db2cai 38> db2audit archive database CAI to /home/db2inst1/sqllib/security/audarchive

Or

db2audit archive to /home/db2inst1/sqllib/security/audarchive

Again, this is not suggested as anything but an emergency solution.

#### Extract Archived Audit Log Files to Comma-delimited Files

jq03a010:db2cai 39> db2audit extract delasc to /db2/CAD/AUDIT/audarchive from files db2audit.db.CAD.log.0.20250124230432

You can choose whether to archive instance or database archive logs or even both.

As a practical matter, I suggest “wild carding” the database and archive files. This will extract to the comma-delimited files in the audarchive subdirectory and it always appends to these files. If you are extracting for a request on a specific time period, I suggest moving, deleting or nulling out the existing files.

As stated earlier, the best candidate for a single source of truth should be the archived audit logs.   
  
jq03a010:db2cai 39> db2audit extract delasc to /db2/CAD/AUDIT/audarchive from files db2audit.db.CAD.log.0.2025012\*

jq03a010:db2cai 40> db2audit extract delasc to /db2/CAD/AUDIT/audarchive from files db2audit.instance.log.0.2025012\*

#### Loading files to tables

Once again, ensure the [tablespace status](#_6pvgtpavis9w) of your tablespaces is 0X0000 before loads.

The “auditlobs” file generated by the “db2audit extract” command will require special handling. It is used with the “execute.del” and context.del” files to get the detailed data from items like the SQL statements.

Proof?   
jq03a010:db2cai 41> strings execute.del | grep auditlob

",2,6342,0,0,"READ\_DML","auditlobs.6590755.87/","CS","auditlobs.6590842.912/"

The "auditlobs.6590755.87/","CS","auditlobs.6590842.912/" is a pointer to the location in the file.   
  
So let’s take some examples:  
  
1. Unless you have a specific reason, always use “**nonrecoverable”** to avoid [LOAD PENDING](#_ps33p3t5admj) States.

2. For the execute and context tables: always use the “LOBSINFILE” keyword.   
  
jq03a010:db2cai 55> db2 "LOAD FROM /db2/CAI/AUDIT/audarchive/execute.del OF del MODIFIED BY **LOBSINFILE** INSERT INTO **audit.execute** **nonrecoverable**";

jq03a010:db2cai 55> db2 "LOAD FROM /db2/CAI/AUDIT/audarchive/context.del OF del MODIFIED BY **LOBSINFILE** INSERT INTO **audit.execute** **nonrecoverable**";  
  
For the others files (objmaint.del, secmaint.del, validate.del, checking.del. Audit.del and sysadmin.del):

jq03a010:db2cai 55> db2 "LOAD FROM /db2/CAI/AUDIT/audarchive/**sysadmin.del** OF del MODIFIED BY **LOBSINFILE** INSERT INTO **audit.execute** **nonrecoverable**";

#### SQL Queries for Audit Tables

These are just some samples, I have zero specifications of what is wanted.

Let’s worry about what the auditors want in this specific case.

Data for the SAPCAP user. We get a lot of entries for the SYSTOOLS.HMON\* and want to filter them out.

jq03a010:db2cai 56> db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10) AS USER, \

SUBSTR(AUTHID,1,10) as AUTHID, SUBSTR(STMTTEXT,1,110) AS SQL FROM AUDIT.EXECUTE \

where UPPER(substr(STMTTEXT,1,6)) in ('UPDATE','INSERT','DELETE') \

and AUTHID = 'SAPCAP' and STMTTEXT NOT LIKE '%SYSTOOLS.HMON%' and STMTTEXT NOT LIKE '%SESSION.%'"

Note the use of the UPPER clause. Without a computed column or a function/expression based index, you would have to account for mixed case SELECT, DELETE, INSERT and UPDATE keywords. Indexes on STMTTEXT won’t really work because of the column size.   
  
Here’s some random samples for the other tables, adapt as needed:  
  
db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10)AS USER, SUBSTR(STMTTEXT,1,110) AS SQL FROM AUDIT.context WHERE USERID != 'hvrcai' and STMTTEXT != '- '"

The remaining tables

db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10) AS USER, SUBSTR(AUTHID, 1, 10) AS AUTHID,CATEGORY, EVENT FROM AUDIT.AUDIT WHERE USER IN ('uskmf90', 'usdjv01')"

db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10) AS USER, SUBSTR(AUTHID, 1, 10) AS AUTHID, CATEGORY, EVENT FROM AUDIT.SYSADMIN WHERE USER IN ('uskmf90', 'usdjv01')"

db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10) AS USER, SUBSTR(AUTHID, 1, 10) AS AUTHID, CATEGORY, EVENT FROM AUDIT.CHECKING WHERE USER IN ('uskmf90', 'usdjv01')"

db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10) AS USER, SUBSTR(AUTHID, 1, 10) AS AUTHID, CATEGORY, EVENT FROM AUDIT.OBJMAINT WHERE USER IN ('uskmf90', 'usdjv01')"

db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10) AS USER, SUBSTR(AUTHID, 1, 10) AS AUTHID, CATEGORY, EVENT FROM AUDIT.VALIDATE WHERE USER IN ('uskmf90', 'usdjv01')"

db2 "SELECT TIMESTAMP, SUBSTR(USERID, 1, 10) AS USER, SUBSTR(AUTHID, 1, 10) AS AUTHID, CATEGORY, EVENT FROM AUDIT.SECMAINT WHERE USERID IN ('uskmf90', 'usdjv01')"

### SAP, IBM and other resources

#### 

#### Db2audit shows unexpected SYSTOOLS.HMON\_ATM\_INFO data

<https://www.ibm.com/support/pages/db2audit-shows-unexpected-systoolshmonatminfo-data>

### 

Problem

db2audit output shows several unexpected SYSTOOLS.HMON\_ATM\_INFO related data. An example is below: SELECT STATS\_LOCK, REORG\_LOCK FROM SYSTOOLS.HMON\_ATM\_INFO WHERE SCHEMA = ? AND NAME = ? AND CREATE\_TIME = ? FOR UPDATE

Cause

The creation of these Health Monitor entries as part of the db2audit output can't be hindered. When policies apply to a user that itself has a particular authority (f.i. SYSADM) as the db2hmon EDU is running under this user, then the SYSTOOLS.HMON\_ATM\_INFO data will be reported as part of the audit log events.

Resolving The Problem

Related Information

[Db2audit](http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.db2.luw.admin.sec.doc/doc/c0005483.html)

#### OSS Note 1474404 - DB6: Poor database performance with DB2AUDIT

<https://me.sap.com/notes/1474404>  
  
Symptom

You experience poor database performance after activating the Db2 audit facility (db2audit). Depending on the nature and number of audit events generated, you notice a poor performance for monitored objects, for example, long insert, update, or delete times.

Other Terms

DB6, DB2 for LUW, audit\_buf\_sz, db2audit

Reason and Prerequisites

The Db2 audit facility generates and allows you to maintain an audit trail for a series of predefined database events. The records that are generated from this facility are kept in an audit log file. The analysis of these records can reveal usage patterns that indicate system misuse by certain users or user groups. Once such system misuse has been identified, you can take actions to reduce or eliminate it.

Database performance may be impacted due to the amount of records generated and the way the records are written to disk.   
  
When events occur, the audit records can be written synchronously or asynchronously to the active log. The value of the AUDIT\_BUF\_SZ database manager configuration parameter determines how the audit records are written.  
  
If the value of AUDIT\_BUF\_SZ is zero (0), the audit records are written synchronously and the audit buffer is not used. The event that generates the audit record waits until the record is written to disk. The wait that is associated with each record decreases the performance of the Db2 database.  
  
If the value is larger than zero (0), space is allocated for the audit buffer (which is part of the database manager shared memory) where the audit records will be placed when they are generated by the audit facility.  
  
If you change this parameter from the default to a value larger than zero (0), the audit facility writes records to disk asynchronously while the audit records are generated from the events, which improves the Db2 performance.

The db2audit categories “execute” and “context” log all SQL statements in the database. If you use the audit categories “execute” and “context”, the respective db2audit log files can become very large, resulting in a negative impact on system performance.

Solution

You can get the current value of database manager parameter AUDIT\_BUF\_SZ using the following command:  
  
*db2pd -dbm | grep AUDIT\_BUF\_SZ*The default value of database manager parameter AUDIT\_BUF\_SZ is 0. If you use the Db2 audit facility, we recommend that you set the initial value of AUDIT\_BUF\_SZ to 4096 (4 KB pages) to allow for asynchronous auditing using the following command:  
  
*db2 "update DBM CFG using AUDIT\_BUF\_SZ <size\_in\_4KB-pages>"*Depending on the number of audit events generated and the number of objects audited, you might have to increase this buffer to improve performance.

To minimize the amount of audit log records written, do not use the category “context”. To disable db2audit for this category on instance level, use the following command:

*db2audit configure scope context status none*

If you use the recommended db2audit setup on database level, do not create db2 audit policies with the categories “execute” or “context”, or limit the use of those categories to selected objects only.

**Note:**The audit buffer cannot be allocated dynamically. Therefore, you must restart the database manager before the new value for this parameter takes effect.  
  
You can monitor the time an agent spends waiting to open and write an audit event synchronously to disk, and also the time that is spent waiting for space in the audit buffer.  
These two monitor elements are as follows:  
  
AUDIT\_SUBSYSTEM\_WAIT\_TIME - Audit file write wait time monitor element  
For more information, see the IBM Knowledge Center at <https://www.ibm.com/support/knowledgecenter/SSEPGG_11.5.0/com.ibm.db2.luw.admin.mon.doc/doc/r0054015.html>

AUDIT\_FILE\_WRITE\_WAIT\_TIME - Audit subsystem wait time monitor element  
For more information, see the IBM Knowledge Center at <https://www.ibm.com/support/knowledgecenter/SSEPGG_11.5.0/com.ibm.db2.luw.admin.mon.doc/doc/r0054013.html>  
  
The above time spent metrics allow you to identify if the audit buffer has been adequately sized for asynchronous auditing and if the operating system write performance and disk performance are sufficient.

#### Load Pending State

##### How to reset LOAD PENDING state

I cannot access a table due to SQL0668N error RC=3, which indicates the table is in LOAD PENDING state. How can I reset the LOAD PENDING state ?

SQL0668N Operation not allowed for reason code "3" on table "<table-name>".

Cause

The table may be placed in LOAD PENDING state if the LOAD operation for the table failed or was terminated in the middle. For example, due to FORCE APPLICATION issued for the application running the LOAD operation.

Answer

You can reset the LOAD PENDING state by any of the following methods.

Terminate the LOAD operation by specifying the TERMINATE option in the LOAD command.

The target table name must match the one specified in the original LOAD command invocation.

db2 "load from test.del of del messages msg.txt terminate into tab1"

Restart the LOAD operation by specifying the RESTART option in the LOAD command.

db2 "load from test.del of del messages msg.txt restart into tab1"

Run a LOAD REPLACE operation for the table in LOAD PENDING state.

db2 "load from test.del of del messages msg.txt replace into tab1 copy yes to /work"

Restore the tablespace containing the table from the latest database or tablespace backup image, then roll forward to the point you would like to recover.

db2 "restore db sample from /work taken at 20110815131050"

db2 rollforward db sample to end of logs and stop

You (also) can find out which table is in LOAD PENDING state by using ADMINTABINFO administration view.

**db2 "SELECT VARCHAR(TABSCHEMA,30) TABSCHEMA,VARCHAR(TABNAME,30) TABNAME, LOAD\_STATUS, NO\_LOAD\_RESTART FROM SYSIBMADM.ADMINTABINFO WHERE LOAD\_STATUS <> 'NULL'"**

TABSCHEMA TABNAME LOAD\_STATUS NO\_LOAD\_RESTART

----------- ---------- ------------ ---------------

DB2INST1 TAB1 PENDING N

You should see this:  
  
jq03a010:db2cai 21> db2 "SELECT VARCHAR(TABSCHEMA,30) TABSCHEMA,VARCHAR(TABNAME,30) TABNAME, LOAD\_STATUS, NO\_LOAD\_RESTART FROM SYSIBMADM.ADMINTABINFO WHERE LOAD\_STATUS <> 'NULL'"

TABSCHEMA TABNAME LOAD\_STATUS NO\_LOAD\_RESTART

------------------------------ ------------------------------ ------------ ---------------

0 record(s) selected.

It’s very slow and I suggest the simple   
  
db2 list tablespaces |grep -i state | grep -v 0x0000

If there’s output that’s non-zero, the likely causes are a backup, restore or load pending state on one of the tablespaces.

#### [db2audit extract returns AUD0036N](https://www.ibm.com/support/pages/db2audit-extract-returns-aud0036n)

<https://www.ibm.com/support/pages/db2audit-extract-returns-aud0036n>

Basically, you are trying to extract files from the active audit log rather than the archived logs. .

Problem

Extraction of audit log using db2audit utility returns, AUD0036N Extract can not be performed on the active audit log file "file-name"

Symptom

This could be demonstrated as follows:

$ db2 "CREATE AUDIT POLICY SENSITIVEDATAPOLICY CATEGORIES EXECUTE STATUS BOTH ERROR TYPE AUDIT"

DB20000I The SQL command completed successfully.

$ db2 "AUDIT TABLE EMPLOYEE USING POLICY SENSITIVEDATAPOLICY"

DB20000I The SQL command completed successfully.

$ db2 "select \* from employee"

$ ls /home/db2inst1/sqllib/security/auditdata

db2audit.db.SAMPLE.log.0

$ db2audit extract file report.out from files /home/db2inst1/sqllib/security/auditdata/db2audit.db.SAMPLE.log.0

AUD0036N Extract can not be performed on the active audit log file "/home/db2inst1/sqllib/security/auditdata/db2audit.db.SAMPLE.log.0".

AUD0001N Operation failed.

Cause

The audit log file has not been archived and hence is still active.

Resolving The Problem

The problem may be resolved as follows:

$ mkdir /home/db2inst1/sqllib/security/audarchive

$ db2audit archive database sample to /home/db2inst1/sqllib/security/audarchive

Node AUD Archived or Interim Log File

Message

-------- -------- ---------------------------------------------------

0 AUD0000I db2audit.db.SAMPLE.log.0.20130402014335

AUD0000I Operation succeeded.

$ db2audit extract file report.out from files /home/db2inst1/sqllib/security/audarchive/db2audit.db.SAMPLE.log.0.20130402014335

AUD0000I Operation succeeded.

#### Helpful sites:

#### [Using db2audit with SAP Applications - Website and Tutorial](https://community.sap.com/t5/technology-blogs-by-sap/using-db2audit-with-sap-applications/ba-p/13445641)

<https://community.sap.com/t5/technology-blogs-by-sap/using-db2audit-with-sap-applications/ba-p/13445641>

### OSS Notes

#### 1474404 - DB6: Poor database performance with DB2AUDIT

<https://me.sap.com/notes/1474404/E>

Symptom

You experience poor database performance after activating the Db2 audit facility (db2audit). Depending on the nature and number of audit events generated, you notice a poor performance for monitored objects, for example, long insert, update, or delete times.

Other Terms

DB6, DB2 for LUW, audit\_buf\_sz, db2audit

Reason and Prerequisites

The Db2 audit facility generates and allows you to maintain an audit trail for a series of predefined database events. The records that are generated from this facility are kept in an audit log file. The analysis of these records can reveal usage patterns that indicate system misuse by certain users or user groups. Once such system misuse has been identified, you can take actions to reduce or eliminate it.

Database performance may be impacted due to the amount of records generated and the way the records are written to disk.   
  
When events occur, the audit records can be written synchronously or asynchronously to the active log. The value of the AUDIT\_BUF\_SZ database manager configuration parameter determines how the audit records are written.  
scope  
If the value of AUDIT\_BUF\_SZ is zero (0), the audit records are written synchronously and the audit buffer is not used. The event that generates the audit record waits until the record is written to disk. The wait that is associated with each record decreases the performance of the Db2 database.  
  
If the value is larger than zero (0), space is allocated for the audit buffer (which is part of the database manager shared memory) where the audit records will be placed when they are generated by the audit facility.  
  
If you change this parameter from the default to a value larger than zero (0), the audit facility writes records to disk asynchronously while the audit records are generated from the events, which improves the Db2 performance.

The db2audit categories “execute” and “context” log all SQL statements in the database. If you use the audit categories “execute” and “context”, the respective db2audit log files can become very large, resulting in a negative impact on system performance.

Solution

You can get the current value of database manager parameter AUDIT\_BUF\_SZ using the following command:  
  
jq03a010:db2cai 21> *db2pd -dbm | grep AUDIT\_BUF\_SZ*The default value of database manager parameter AUDIT\_BUF\_SZ is 0. If you use the Db2 audit facility, we recommend that you set the initial value of AUDIT\_BUF\_SZ to 4096 (4 KB pages) to allow for asynchronous auditing using the following command:  
  
jq03a010:db2cai 21> *db2 "update DBM CFG using AUDIT\_BUF\_SZ <size\_in\_4KB-pages>"*Depending on the number of audit events generated and the number of objects audited, you might have to increase this buffer to improve performance.

To minimize the amount of audit log records written, do not use the category “context”. To disable db2audit for this category on instance level, use the following command:

jq03a010:db2cai 21> *db2audit configure scope context status none*

If you use the recommended db2audit setup on database level, do not create db2 audit policies with the categories “execute” or “context”, or limit the use of those categories to selected objects only.

**Note:**The audit buffer cannot be allocated dynamically. Therefore, you must restart the database manager before the new value for this parameter takes effect.  
  
You can monitor the time an agent spends waiting to open and write an audit event synchronously to disk, and also the time that is spent waiting for space in the audit buffer.  
These two monitor elements are as follows:  
  
AUDIT\_SUBSYSTEM\_WAIT\_TIME - Audit file write wait time monitor element  
For more information, see the IBM Knowledge Center at <https://www.ibm.com/support/knowledgecenter/SSEPGG_11.5.0/com.ibm.db2.luw.admin.mon.doc/doc/r0054015.html>.

AUDIT\_FILE\_WRITE\_WAIT\_TIME - Audit subsystem wait time monitor element  
For more information, see the IBM Knowledge Center at <https://www.ibm.com/support/knowledgecenter/SSEPGG_11.5.0/com.ibm.db2.luw.admin.mon.doc/doc/r0054013.html>.  
  
The above time spent metrics allow you to identify if the audit buffer has been adequately sized for asynchronous auditing and if the operating system write performance and disk performance are sufficient.

db2 "SELECT application\_handle, rows\_returned, tcpip\_send\_volume, \

AUDIT\_SUBSYSTEM\_WAIT\_TIME, AUDIT\_FILE\_WRITE\_WAIT\_TIME \

FROM TABLE(MON\_GET\_CONNECTION(cast(NULL as bigint), -2)) AS t \

ORDER BY rows\_returned DESC"

APPLICATION\_HANDLE ROWS\_RETURNED TCPIP\_SEND\_VOLUME AUDIT\_SUBSYSTEM\_WAIT\_TIME AUDIT\_FILE\_WRITE\_WAIT\_TIME

-------------------- -------------------- -------------------- ------------------------- --------------------------

192 117480239 17407440544 0 0

188 114086412 29296915100 0 0

239 111656170 23085287052 0 0

197 108719246 14454980556 0 0

206 101932041 22608094042 0 0

218 96007302 23190931334 0 0

