SQL Auditing Facility

User's Guide

Software
Product
Research

SQL/Auditing Facility for DB2/VSE
Version 1.1

1 Functional Description

1.1 Auditing Access

The SQL/Auditing Facility - called "**SQL/AF"** from now on - records user and program access to designated DB2/VSE tables in databases managed by a DB2/VSE server. The Auditing Facility is offered as a chargeable feature of the SQL/Monitoring Facility program product.

The product performs its auditing functions in the partition of the database server. Therefore, it is capable of auditing VSE/ESA and CICS clients, as well as non-DB2/VSE clients accessing the DB2/VSE database through IBM's DRDA protocol. One SQL/AF component (the *Audit Initiator*) executes in the database server, another component (the *Audit Processor*) executes in the SQLAFP service partition. The Audit Processor writes to the audit dataset. The Initiator and the Processor components exchange data using an SQL/AF dataspace.

SQL/AF monitors access to all tables defined as candidates for auditing in the **SQLAF RULES** dataset. All table access, both dynamic (using QMF, ISQL etc.) and static (compiled) is subject to auditing. Depending on the audit rules defined for the table, both read (SQL SELECT) and write (SQL DELETE, INSERT and UPDATE) access is monitored. By default, all table columns are audited so that auditing occurs for all SQL statements accessing the table. Alternatively, one or more table-columns can be defined in the **RULES** dataset: auditing then occurs only when an SQL statement refers to one of these protected table columns.

SQL/AF also monitors DB2/VSE **prep** for packages that access audited tables. This will effectively result in the recording of a **change history** for such packages.

For each SQL statement accessing a protected table or table-column, the Audit Processor maintains following data in the **SQL/AF audit log**

- the date and time of statement execution
- the VSE and DB2 name of the user executing the statement
- the identification of the terminal from which the statement has been executed (to obtain the CICS terminal number, DB2/VSE 3.5 or higher must be installed)
- the name of the package (program) containing the statement
- the statement type as: SELECT, UPDATE, INSERT, DELETE, COMMIT, ROLLBACK, PREP
- the number of rows affected by the statement
- the LUW status for the statement (committed or rolled-back)
- the "SQLCODE" associated with the statement[1]
- the text of the executed SQL statement[2].

[1]The primary purpose is to audit illegal attempts to access protected data (SQLCODE -551).

[2]The text of compiled statements is obtained from the SQLCS_SQL_STMNTS table, which is maintained by the SQL/MF program product.

Following transformations are performed on the text of the statement

- the statement text is shown as it is actually executed by DB2/VSE, that is, with all host variable references replaced by the **contents** of the variables[3]
- if the statement accesses an audited table through a view, all view references are replaced with their real table equivalent
- *special register* references (such as CURRENT DATE and CURRENT TIMESTAMP) are replaced with the actual value of the register
- if an INSERT statement does not use a column-list, a default list with all table column names is generated
- INSERT format-2 statements are replaced by INSERT format-1[4]
- cursor-based PUT statements are replaced with an INSERT
- UPDATE or DELETE statements using the WHERE CURRENT OF CURSOR clause are transformed in such a way, that the column names and column values obtained during the last fetch on the current cursor substitute the WHERE CURRENT OF clause

A statement is audited when it references or modifies one of the protected table columns, as follows:

- SELECT statements are audited when a protected table column (explicitly or implicitly) appears in the SELECT column-list or in the SELECT WHERE clause. For cursor-based SELECTs, auditing is performed when the SELECT cursor is opened and closed[5].
- All DELETE statements on a protected table are audited. Protected columns are not considered during delete auditing.
- UPDATE statements are audited when one or more protected columns appear in the UPDATE SET or WHERE clauses.
- INSERT or PUT statements are audited when one or more protected columns are inserted by the statement.
- If a table has been defined in SQLAF RULES without a column list, **all** its columns are protected.

Audit events are stored while the originating application is executing. If the application is rolled back, either implicitly or explicitly, the update, insert or delete events may have been audited already. Therefore, the audit log contains a COMMIT or ROLLBACK entry stored at the end of each logical unit of work. This allows the auditor to mark the table accesses as effective (COMMIT) or non-productive (ROLLBACK).

1.2 Audit Log Datasets

During normal operation, the audit processor writes the audit output to the **primary audit log** dataset. When archiving the primary log to tape, incoming audit request are temporarily stored on a **secondary audit log** dataset. The audit logs are stored as partitioned datasets in the **SPRLIB** VSAM dataset, using a logical access method provided by SQL/AF. This access method uses a VSAM KSDS as the underlying physical access method.

[3]If a host variable assigns a LONG VARCHAR column, only the first 32 characters of the substitution value are inserted.

[4]This is the only statement for which the Auditor accesses the DB2/VSE log.

[5]FETCH statements are not logged. However, when closing the cursor, the number of rows fetched is inserted in the audit log.

1.3 Log Archiving

The SQL/AF **archiving** function transfers the **primary audit log** from disk to an archive tape, so that auditing results can be kept for a longer period of time. Archiving must be scheduled explicitly by the user.

1.4 Inspecting the Audit Log

The SQL/AF user interface program $AFS is a CICS transaction. It allows to interactively search the audit log for specific audit events. To perform the log scan, the user supplies a number of search criteria.

Following items may be used as search criteria:

- One or more fields of the audit log record.

- The column names and column values in the audited SQL statement text. This powerful scan method examines the text of the SQL statements found in the audit log and selects log entries:

  - where the statement text references a named table column
  - where the statement text references a named table column with a specified value

Sample search criteria

- search all accesses made by a named user to a named table during a specified period
- search all updates made by a named program to a named table on a specified date
- search all updates made to the EMPLOYEE.SALARY column during a specified period
- search all accesses made to the EMPLOYEE table for EMPNO 100 during a specified period
- search all updates that make the CUSTOMER.BALANCE column negative

1.5 Statement text caching

Before storing the audit log record for a static (compiled) statement, the text of the statement is retrieved by the Audit Processor from the monitor table SQLCS_SQL_STMNTS. To avoid continuous SELECTs on this table, all statement texts retrieved are maintained in **cache** storage. Enough virtual storage should be provided to the Audit Processor for implementing this cache.[6] However, if a storage constraint exists, the Audit Processor may be requested to remove cached statements that have not been used during a specified period of time.

[6] If the average length of the statement text, before hostvariable substitution, is 256 characters, 1 Mb allows to cache about 3500 SQL statements.

2 Installing the Audit Facility

2.1 Software Prerequisites

- VSE/ESA Version 2 Release 2 and later
- DB2/VSE Version 3.5 and later
- SQL/MF Version 1 or later (SQL/MF is a product from Software Product Research)

2.2 Installing SQL/AF

The SQL/AF software is delivered as a PC file in ZIP format.

Place the ZIP file in a dedicated directory (e.g. SQLAF) and unzip the file. Following files should now be present in the SQLAF directory:

INSTALL.BAT	installation procedure for Windows
SEND2RDR.BAT	installation procedure for Windows
SQLAF.PCF	SQL/AF software
SQLAFSCF.PCF	software key
SQLAFI0.VSEJOB	setparm SPRLIB
SQLAFI1.VSEJOB	linkedit SQL/AF
SQLAFI2.VSEJOB	define the SQL/AF report library
SQLAFI3.VSEJOB	define the SQL/AF components to CICS
SQLAFI4.VSEJOB	install SQL/AF in a database
SQLAFI5.VSEJOB	define the audit log
SQLAFP.VSEJOB	job to start the SQL/AF server
SQLAFPR.VSEJOB	job to restart the SQL/AF server
SQLAF.BKS	bookshelf for IBM Library Reader
SQLAF.BOO	SQL/AF User's Guide in IBM Library Reader format
SQLAF.PDF	SQL/AF User's Guide in PDF format

2.2.1 Preliminary Note

The INSTALL.BAT and SEND2RDR.BAT installation procedures use the “SEND.EXE” to upload the PC files to the POWER reader queue. Ensure that your 3270 emulator supports SEND (some emulators don’t) and that the EXE is on your active path.

If you cannot use the SEND.EXE, use the upload facilities of your emulator to perform the equivalent of the SEND commands contained in the BAT files, that is:

for the INSTALL.BAT:

```
SEND SQLAFSCF.PCF (FILE=RDR BINARY LRECL=80 NOUC
SEND SQLAF.PCF (FILE=RDR BINARY LRECL=80 NOUC
```

for the SEND2RDR.BAT:

```
SEND <filename> (FILE=RDR
```

2.2.2 Upload the SQL/AF software

- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Execute the INSTALL.BAT by clicking. This will upload 2 jobs to the POWER/VSE reader queue with DISP D and class 0. The jobs catalog all SQL/AF components into the VSE library previously chosen as residence for the prerequisite product SQL/MF
- After job completion, check the DISP=H listing for any errors.

2.2.3 Create the SQL/AF SVA List $SVASQLA

- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Drag and drop the file SQLAFI1.VSEJOB on the SEND2RDR.BAT. This will send and start the job in class 0 and DISP D. The output listing is in DISP H.
- Check the output listing for errors.

2.2.4 Link SQL/AF

- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Drag and drop the file SQLAFI2.VSEJOB on the SEND2RDR.BAT. This will send and start the job in class 0 and DISP D. The output listing is in DISP H.
- Check the output listing for errors.

2.2.5 Define the SQL/AF components to CICS

- The SQLAFI3.VSEJOB assumes that the CICS.CSD cluster is in the VSESPUC user catalog. If this not not the case, re-assign the CAT symbol in the SQLAFI3.VSEJOB file.
- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Drag and drop the file SQLAFI3.VSEJOB on the SEND2RDR.BAT. This will send and start the job in class 0 and DISP D. The output listing is in DISP H.
- The job starts a DFHCSDUP run that defines the interactive SQL/AF components to CICS, using the group SQLAF.
- Check the output listing for errors.

2.2.6 Install SQL/AF in a database

This installation step should be executed in all databases where SQL/AF will be used. This step will load the SQL/AF packages into the database.

Edit the PC file SQLAFI4.VSEJOB and assign the following SETPARM symbols

CAT
the VSAM catalog for a SAM ESDS workfile (default is VSESPUC)
DB
The name of the target database.
DBAPASS
The password of user SQLDBA in that database.

Submit the SQLAFI4.VSEJOB as follows:

- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Drag and drop the file SQLAFI4.VSEJOB on the SEND2RDR.BAT. This will send and start the job in class 0 and DISP D. The output listing is in DISP H.
- Check the output listing for errors.

2.2.7 Define the SQL/AF Audit Log

- Edit the SQLAFI5.VSEJOB and assign the SETPARM symbols for the job that will create the audit log cluster SPR.SQLAF.LOG. To determine the size of the audit log, please refer to ***SQL/AF Space Estimates*** on page 33.
- Assign following parameters:
  - CAT: the user catalog for the SPR.SQLAF.LOG cluster
  - VOLUME: the volume-ID for the CLUSTER DEFINE command
  - PALLOC: the primary space allocation as a number of records (each record is a 16K block)
  - SALLOC: the secondary space allocation as a number of records (each record is a 16K block)
- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Drag and drop the file SQLAFI5.VSEJOB on the SEND2RDR.BAT. This will send and start the job in class 0 and DISP D. The output listing is in DISP H.
- Check the output listing for errors.

Insert following DLBL in the STDLABUP.PROC

// DLBL SQLALOG,'SPR.SQLAF.LOG',,VSAM,CAT=xxxxxx,DISP=(OLD,KEEP)

2.2.8 Update the CICS FCT

Insert a **COPY SQLAFFCT** in DFHFCT and assembling it. SQLAFFCT.A is in the SQL/AF library. The CICS definition is needed to access the audit log from the interactive log scan transaction **$AFS**.

2.2.9 Upload the SQLAFP startup job

- The SQLAFP VSEJOB starts the SQL/AF server partition.
- Examine the SQLAFP.VSEJOB and modify the CLASS operand, if needed.
- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Drag and drop the file SQLAFP.VSEJOB on the SEND2RDR.BAT. This will store the job in the POWER reader queue, with DISP L.

2.2.10 Upload the SQLAFPR restart job

- Examine the SQLAFPR.VSEJOB and modify the CLASS operand, if needed.
- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Drag and drop the file SQLAFPR.VSEJOB on the SEND2RDR.BAT. This will store the job in the POWER reader queue, with DISP L.

3 Setting up the Audit Facility

3.1 Auditing files

During auditing following files are written:

1. The primary audit log dataset SQLAF AUDIT1. It is written to SPR.AUDIT.LOG and contains the audit output.
2. The secondary audit log dataset SQLAF AUDIT2. It is written to the SPR PDS and temporarily holds audit output while the primary log dataset is being archived to tape.
3. The audit control datasets. They contain DB2 catalog information about the tables and tablecolumns being audited.

To determine the size of the above files, please refer to ***SQL/AF Space Estimates*** on page 33. Most of the allocated space will be taken by the primary log dataset.

3.2 The Audit Processor

The Audit Processor program runs in a static or dynamic partition. The SQLAFP VSEJOB, delivered with the product should be loaded into the VSE/POWER RDR queue. Modify the POWER JOB statement to suit your installation standards. Do not modify the POWER or VSE jobname.

A partition size of 4 megabytes should be sufficient for most installations.

Following startup parameters can be specified on the PARM operand of the EXEC SQLAFP statement:

DSPOLL x Specifies the poll interval of the auditing dataspace. The default poll interval is 3 seconds. Decrease this value on busy systems. Alternatively, increase the size of the audit dataspace by specifying a larger DSPSIZE on the AUDITOR statement (see page 17)

CLEANUP x For performance reasons, the Audit Processor maintains a cache of the SQL statement texts inserted into the Audit Log. This cache is never cleared. If you want to purge the cache periodically, specify a CLEANUP parameter in minutes. You can also specify a larger partition GETVIS to increase the caching capacity.

Specifying an EXEC SQLAFP without PARM is equivalent to:

```
// EXEC SQLAFP,PARM='DSPOLL=3,CLEANUP=0'
```

4 Auditing rules and options

4.1 Defining auditing rules

The tables eligible for auditing (the *protected tables*) and the degree of auditing required for each of them, are defined in the **SQLAF RULES** file, which is a VSE dataset to be setup by the user. The dataset should be located in the SQL/AF VSE library.

Following auditing control statements may be specified in the rules dataset:

PERIOD FROM hh:mm **TO** hh:mm
IGNORE [**VSEID** name] [**SQLID** name] [**PACKAGE** creator.name]
DATABASE name
TABLE creator.name
COLUMN {***** | name}
AUDIT [SELECT] [INSERT] [UPDATE] [DELETE] [ALL] [CHANGE]

Coding Rules

1. The **DATABASE** and **TABLE** are the only required statements. All others are allowed to default.
2. All statements, except **PERIOD** and **IGNORE**, are database-specific and apply to the database named in the last DATABASE statement.
3. When specified **before** the first DATABASE statement, the **PERIOD** and **IGNORE** statements are **global**, that is, they concern all databases named subsequently. When specified **after** a DATABASE statement, the PERIOD and IGNORE statements apply to that database only. In the latter case, different PERIOD and IGNORE statements may be coded for different databases.
4. The **TABLE** and the **AUDIT** statement are specified once for each table to be audited.
5. The **COLUMN** statement may be specified multiple times per table, if multiple table columns should trigger auditing.

With the exception of the PERIOD and IGNORE statements, which are handled by the Audit Initiator, changes to the RULES dataset take effect when the Audit Processor machine is [automatically] restarted.

4.1.1 Defining an auditing period

Syntax : PERIOD FROM hh:mm **TO** hh:mm

Restricts auditing to the defined time period. If omitted, auditing is done without restriction. Please note that the SQL/AF components are called by the SQL/Monitoring Facility. If the SQL/MF **MONPERIOD** statement from the **SQLMF CONFIG** dataset is in effect, auditing will be suspended during the monitor's inactivity.

4.1.2 Specifying audit ignore rules

Syntax : **IGNORE** [**VSEID** name] [**SQLID** name] [**PACKAGE** creator.name]

Designates the VSEIDs, SQLIDs and/or packages for which auditing should be bypassed. The purpose of this command is to prevent auditing during administrative table access such as backup or reorganization. The statement can also be used to bypass auditing for long-running "batch" update jobs. Multiple IGNORE statements can designate multiple combinations of (VSEID, SQLID, Packagename).

If one of the keywords VSEID, SQLID or PACKAGE name is omitted, the item is not checked during ignore. However, at least one of the above keywords must be specified on the IGNORE statement.

For example:

IGNORE VSEID dbamaint SQLID sqldba PACKAGE sqldba.aridsql
Disables auditing for DBSU when executed under the specified VSEID and SQLID.

IGNORE VSEID dbamaint PACKAGE sqldba.aridsql
Disables auditing for DBSU when executed under the specified VSEID, whatever the SQLID.

IGNORE SQLID dbamaint
Disables auditing for all packages executed under the specified SQLID, whatever the VSEID.

4.1.3 Specifying the audited database

Syntax : DATABASE name

Designates the database to which all following rule statements apply. It should be specified before all other statements defining audit rules for this database. The current database remains in effect until the next DATABASE statement. All tables audited for a given database should all be specified after the DATABASE statement.

4.1.4 Specifying the audited tablename

Syntax : TABLE creator.name

Designates the creator and the name of the table to be audited. Views on audited tables need not be defined as such, since a view inherits the audited status from its base table.

Generic creator and/or tablenames can be specified by including a **trailing** % sign.[7] The COLUMN and AUDIT statements following a generic TABLE statement, will apply to all the tables satisfying the generic name.[8]

A given table name may be occur multiple times, either implicitly or explicitly, in the SQLAF RULES. For example: global auditing rules can be defined for a set of tables by using a generic name. Subsequently, the auditing rules for individual tables of the set can be redefined. The COLUMN and AUDIT statements specified at the **last** table occurrence will take effect.

4.1.5 Specifying audited column names

Syntax : COLUMN {***** | name}

Designates a protected table column. Access to such a column triggers an audit event. If the statement is omitted, COLUMN * is assumed and all table columns are protected. If several protected columns are to be designated for the same table, a COLUMN statement line must be coded for each of them.

[7]The generic name specifications in the RULES are more restrictive than the usual SQL syntax, because they are applied by SQLAFP. For reasons of performance, SQLAFP issues a single SELECT of all tablenames from SYSCATALOG and performs creator and tablename selection on the obtained list.

[8]System tables (CREATOR=SYSTEM) cannot be included using a generic name specification. To audit catalog access, each catalog table must be specifically named in the rules dataset.

4.1.6 Specifying the audit level

Syntax : AUDIT [SELECT] [INSERT] [UPDATE] [DELETE] [ALL] [CHANGE]

Defines which SQL statements should be audited. The **AUDIT** parameter can have one or more of the following values and triggers the corresponding audit action:

SELECT
Audits SQL SELECT commands. In case of cursor based SELECTs that access multiple rows, only the OPEN and CLOSE of the cursor is audited.

INSERT
Audits SQL INSERT commands.

UPDATE
Audits SQL UPDATE commands.

DELETE
Audits SQL DELETE commands.

ALL
AUDIT ALL is a shorthand for **AUDIT SELECT INSERT UPDATE DELETE**

CHANGE
AUDIT CHANGE is a shorthand for **AUDIT INSERT UPDATE DELETE**

If multiple **AUDIT** keywords are required, they must all be specified on the same **AUDIT** statement. If the statement is omitted, **AUDIT ALL** is assumed.

4.1.7 Comments

A statement starting with the * (asterisk) sign is considered as a comment line.

4.1.8 Sample SQLAF RULES

```
    DATABASE databasename
*
    TABLE SQLDBA.EMPLOYEES   (1)
    COLUMN CONFIDENTIAL_INFO
    AUDIT SELECT
*
    TABLE SQLDBA.SECRET    (2)
*
    TABLE TS.%     (3)
    COLUMN *
    AUDIT CHANGE
*
    TABLE TS.TS1%     (4)
    COLUMN C1
    COLUMN C2
    AUDIT ALL
```

The above auditing rules request:

1. Auditing of all SELECTs on the column CONFIDENTIAL_INFO of the EMPLOYEES table.
2. Auditing of all accesses and changes to any column of the SQLDBA.SECRET table. (COLUMN * and AUDIT ALL clauses are defaulted)
3. Auditing of all changes to any column of the tables created by user TS.
4. Redefines auditing rules for the tables included in (3): for all tablenames starting with the characters TS1, only accesses and modifications to the columns C1 and C2 are audited.

5 Starting the auditor components

5.1 Starting the Audit Initiator

The **audit initiator** program (SQLAFI) runs in the DB2/VSE database server machine. It is automatically initiated by the SQL/Monitoring Facility when the **AUDITOR** statement is found in the **SQLMF CONFIG** dataset for the current database.

The syntax of the AUDITOR statement is as follows:

AUDITOR DSPSIZE n [UNRESTRICTED]

When the UNRESTRICTED operand is not coded:

- A failure during auditing (due to audit processor not logged on, audit queue full etc ...) will logically force the audited user by means of an SQLCODE -901.
- An eventual SQL/MF or SQL/AF abend will force a DB2/VSE shutdown. This is done to prevent users from performing non-audited table access.

When the UNRESTRICTED keyword is coded, none of the above actions are taken and database processing continues without auditing. It is the responsibility of the installation to evaluate the security impact of unrestricted operation.

The AUDITOR statement may be coded in the global SQLMF CONFIG section (in which case all databases in the configuration will be audited) or it may be coded in one or more database **SECTION**s.

5.2 Starting the Audit Processor

The Audit Processor runs in a dedicated machine and is started by including the **SQLAFP** command in the PROFILE EXEC.

The syntax of the SQLAFP command is as follows:

// EXEC SQLAFP, PARM='CLEANUP=n,DSPOLL=n'

CLEANUP

Specify the cache cleanup interval in minutes. Cache cleanup removes the cached command texts of all packages not used during the last (CLEANUP) minutes. If the CLEANUP argument is omitted, no cache cleanup occurs and the text of all audited static SQL statements remains in storage during the entire Audit Processor session. CLEANUP should be requested when not enough GETVIS storage is available to retain all audited commands in the cache.

DSPOLL

This argument specifies the time interval used by the Audit Processor to inspect the dataspaces for auditing requests. This "polling interval" should be expressed in seconds. It defaults to 3 seconds. Specifying larger polling intervals usually requires a larger auditing dataspace because requests stay in the audit queue for a longer period of time.

6 Scanning the Audit Log or Audit Archive

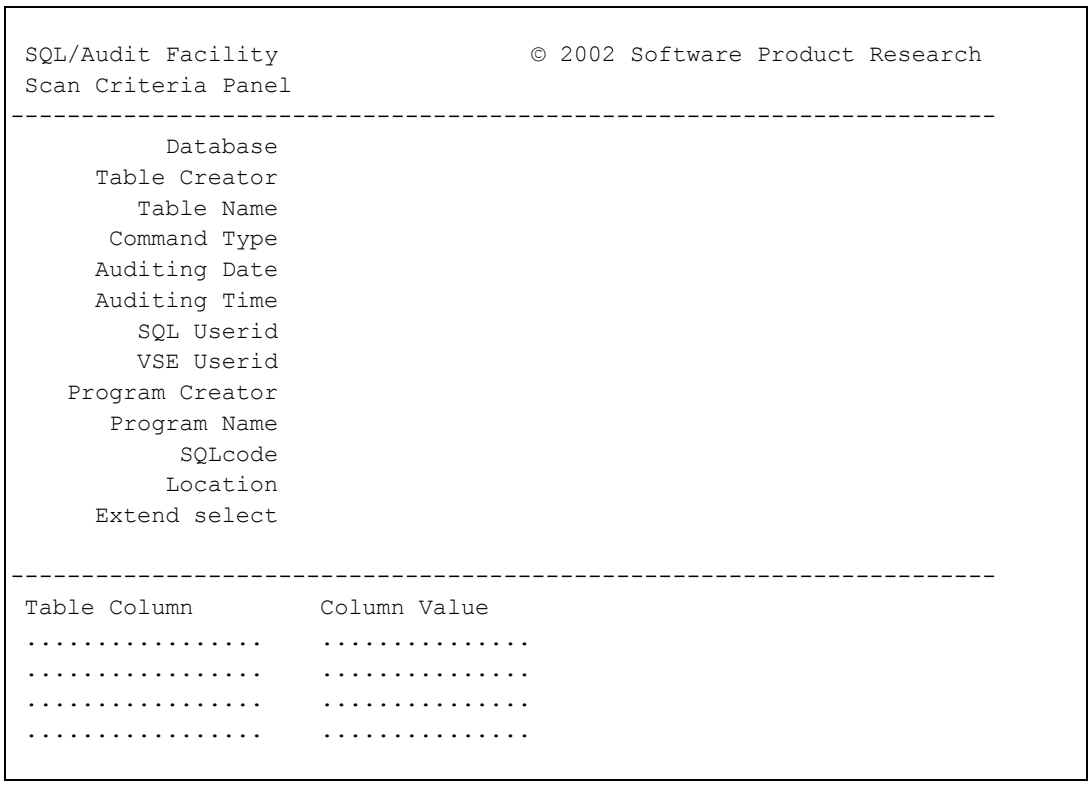
6.1 Scanning the Audit Log

The Log scan program displays selected audit entries from the primary audit log SQLAF AUDIT1. It is invoked as the CICS transaction $AFS.

6.1.1 Specifying scan criteria

At entry into log scan, the program displays the **search criteria panel** for the current log scan. The upper part of the screen allows you to enter the scan values for the **audit record fields**. The bottom lines of the screen are used to enter up to 4 **table column scan criteria** to be applied against the text of the audited SQL statements.

The search criteria specified at the previous invocation of the utility are displayed as current defaults, which you may accept or modify.



Following PFkeys can be used on the above panel:

PF1 Displays the help file.
PF3 Quits the selection panel.

6.1.1.1 Scanning audit log record fields

Following log record fields can be used as search arguments:

- name of database accessed
- table creator and tablename accessed
- command type (to scan access log entries, enter *SELECT, INSERT, UPDATE* or *DELETE*, to display program prep log entries, enter *PREP* as command type)
- date of auditing as *yyyy-mm-dd*
- time of auditing as *hh:mm:ss*
- DB2 userid performing the access
- VSE user name performing the access
- package creator and package name performing the access
- command's SQLCODE as *-nnn*
- user's terminal name or batch jobname

Syntax rules

- When a criteria field is blank, no test is performed on that field during scan.
- By default an "equal" test is performed on criteria fields.
- However, a generic test is possible, as follows:
  - by specifying a trailing % sign, you can scan the leading positions of a record field
  - by specifying a leading % sign, you can test whether the specified scan string occurs in the field
  - by placing one of the logical operators >, <, >=, <=, or <> after the search value, a criteria field can be tested greater than, lower than or not equal.
- Scan values should **not** be enclosed in quotes.
- Multiple field scan criteria may be specified on the same panel. An audit record will be selected only when it satisfies **all** the specified criteria.

EXAMPLES

Program Name ARIISQL selects accesses done by program ISQL
Program Name ARI% selects program names starting with "ARI"
Program Name %SQL selects program names containing "SQL"
Auditing Time 15>= selects auditing times greater than or equal to 15h.

6.1.1.2 Scanning the audited command text

Column-names and column-values appearing in the audited command text may also be used as scan criteria.

If a column-name is specified alone, audited commands are selected if the column-name appears in the command text.

If both a column-name and a column-value are specified, selection will occur if the command text refers to the column-name with the specified value.

Column-name / column-value pairs are recognized as follows:

SELECT, DELETE and UPDATE commands

- The columnname-columnvalue pair is searched in all WHERE predicate expressions having the format <column-name> <operator> <constant-value>. The scan logic examines the predicate operators = , ^=, <>, <, <=, >, >=, LIKE, IN and BETWEEN and uses them in matching the scan criterion.

- For example: the scan criterion **"COL 100"** will be satisfied by the WHERE clauses COL=100, COL<200, COL <> 50 etc..

- However in the present version of the product, ANDed and ORed combinations of WHERE predicate clauses are **not** handled: a scan terminates as soon as a *true* result has been obtained from a predicate clause.

- For example: the search criterion "COL 100" will be satisfied for the predicate *WHERE COL > 50 AND COL < 200* but also for the predicate *WHERE COL > 50 AND COL < 90*, since in both cases the expression COL > 50 terminates the scan. The search criterion "COL 100<" on the other hand, will be satisfied for the predicate *WHERE COL > 50 AND COL < 90* only.

INSERT commands

The column-name is searched in the INSERT column-list and the column-value in the corresponding position of the VALUES clause.

UPDATE commands

In addition to an eventual UPDATE WHERE clause, the SET clauses are examined for "columnname, columnvalue" occurrences.

Syntax rules

- Column character values should **not** be enclosed in quotes.
- Column values that are logically numeric appear in the audited command text as edited character strings and are scanned as such. Therefore, the following rules should be observed when supplying a numeric search value:
  - leading zeroes in numeric values should **not** be specified
  - trailing zeroes of a fractional value may be specified or omitted
  - negative values should start with a **-** sign
  - a decimal point is entered as a period
- You can perform a **generic** test on column values, as follows:
  - by specifying a trailing % sign, you can scan the leading positions of the value
  - by specifying a leading % sign, you can test whether the specified scan string occurs in the column value
  - testing whether a numeric column is negative can be done using the search strings **-%** or **%-** since a negative value is stored in the log as the character string -nnn
- Up to 4 column scan criteria may be specified on the same panel. An audit entry will be selected only when it satisfies all the specified record field and column value criteria.

6.1.2 Processing the scan results

The results of the audit log scan are presented on the screen as a report. A PFkey interface is provided for handling the report.

The default format of the report is as follows:

```
SQL/Auditing Facility - Scan Report                          Page 1 of 1
 ------------------------------------------------------------------------------
TableName              AuditDate  Time     Database SQL_User VSE_User Term
------------------------------------------------------------------------------
SQLDBA.EMPLOYEE        2002-04-29 11:15:06 SPRDB2   METAST2  METAST2  T25T
SQLDBA.EMPLOYEE        2002-04-29 11:15:52 SPRDB2   METAST2  METAST2  T25T
SQLDBA.EMPLOYEE        2002-04-29 11:16:34 SPRDB2   METAST2  METAST2  T25T
SQLDBA.EMPLOYEE        2002-04-29 11:16:43 SPRDB2   METAST2  METAST2  T25T
SQLDBA.EMPLOYEE        2002-04-29 11:18:36 SPRDB2   METAST2  METAST2  T25T
SQLDBA.EMPLOYEE        2002-04-29 11:19:10 SPRDB2   METAST2  METAST2  T25T




 ------------------------------------------------------------------------------
PF  1 Help      2 CommText 3 Quit      4 Hardcopy  5 Format   6 Top
    7 Page <<<  8 Page >>> 9 Bottom   10 View <<< 11 View >>> 12 PageMode
```

Processing the scan report

Following PFkey interface is provided to process the report.

PF1
Displays help.

PF2
Invokes the **Command text** function. This function displays the full and formatted SQL statement text. Long commands may take several pages to display. Use PF7 and PF8 to browse through the pages. Press PF3 to terminate the scan result list.

PF3
Terminates the scan utility.

PF4
Takes a hardcopy of the current report to the VSE/POWER LST queue. The first four characters of the queue entry name are set to 'SQLM'. The last four characters are the name of the requesting terminal.

PF5
Performs the report formatting function.

When invoked, the report format function displays a function menu. Enter the code corresponding to the desired formatting function, as follows:

- **F** reformats the report
- **L** moves the viewing window to the left margin
- **R** moves the viewing window to the right margin

Report reformat

The **F** function shows all columns in the report, preceded by a + sign if the column is currently displayed or by a - sign if the column has been hidden previously. You may override the + or - sign in the following manner:

- Entering **+** unhides a previously hidden column, i.e. the column will be displayed again.
- Entering **-** hides a column so that it is no longer displayed, although it remains in the report.
- Entering **<** sorts the report on this column in ascending sequence (low to high). Only one ordering column can be designated.
- Entering **>** sorts the report on this column in descending sequence (high to low). Only one ordering column can be designated.
- Entering a number from 1 to 9 will move the column to the corresponding position in the report. The column previously on that position will take the position of the column moved.

PF6
Shows the first page of the report.

PF7
Shows the previous page of the report.

PF8
Shows the next page of the report.

PF9
Shows the last page of the report.

PF10
When in list mode, moves the viewing window to the left.
When in page mode, calls the **list search** function. For a description of the search function, see page 34.

PF11
Moves the viewing window to the right.

PF12
Shows the report in “page” mode, as follows:

```
SQL/Auditing Facility - Scan Report                          Page 6 of 6
------------------------------------------------------------------------------
TableName  SQLDBA.EMPLOYEE
AuditDate 2002-04-29          Time   11:19:10        Database   SPRDB2
 SQL_User  METAST2         VSE_User  METAST2         Terminal   T25TT261
   Commit  Y               ProgCrea  SQLDBA          ProgName   ARIISQL
  Section  3               Commtype  UPDATE         Rows_Proc   1
  SQLCODE  0                SQL_LUW  1161165

 ------------------------------------------------------------------------------
UPDATE SQLDBA.EMPLOYEE SET PHONENO = '6912' WHERE EMPNO = '000055'




  ------------------------------------------------------------------------------
 PF  1 Help     2 CommText  3 Quit      4 Hardcopy  5 Format    6 Top
     7 Page <<< 8 Page >>>  9 Bottom   10 Search   11          12 ListMode
```

6.1.3 Searching the scan results

The search function is called by pressing **PF10** when the scan result list is displayed in **page mode.**

The search function displays the following **search criteria panel**:

```
        Search column .......
         Search value .......
    Search direction >
```

Search column
Enter the name of the list column to be searched. Use the column name that appears as column header on the display screen.

Search value
- Enter the value to find in the search column.
- The specified search value can be partial. The search column is examined over the length of the specified value only.
- By default, an **equal** search is performed. By prefixing the search value with one of the logical operators **<, >, <=, >=, ^=, <>** you can perform a not-equal, a lower-than or a greater-than search. Blanks may, but need not, intervene between the operator and the value.
- You can also specify **MIN** or **MAX** as the search value, to find the minimum or maximum value of the search column in the list.

Search direction
Enter the **>** sign to search the list in forward direction, which is the default.
Enter the **<** sign to search the list in backward direction.

Notes

(1) Except for MIN or MAX search, which always scans the entire list, searching starts at the position of the **current report line + 1**. When the search is productive, the current line is positioned on the list entry found. This allows to repeatedly apply the same search arguments.

(2) The last search criteria entered are redisplayed on the next search panel.

(3) Logical operators and MIN/MAX functions can also be applied to non-numerical list columns.

PF key definitions:
PF1 requests help
PF3 exits search

6.2 Scanning an Audit Archive tape

The SQLAFATS program is used to scan one or more SQL/AF audit archive tapes for specific events.

The PARM field on the EXEC SQLAFATS JCL statement specifies the number of archives tapes to be scanned. The syntax is: PARM='TAPES=n' where n designates the number of tapes. Each tape should be identified by a TLBL INxx statement where xx ranges from 01 to 99.

The scan output is stored in the POWER LST queue.

The scan arguments are entered on SYSIPT, following the EXEC SQLAFATS statement. Each argument must be entered on a separate "card".

Following scan arguments are provided:

```
DATABASE N
TABCREA  N
TABNAME  N
CMNDTYP  N
DATE     N
TIME     N
SQLID    N
VSEID    N
PROGCREA N
PROGNAME N
SQLCODE  N
TERMID   N
COLUMN   COLNAME [OPERATOR [VALUE]]
```

The syntax of the scan argument statements is identical to the syntax used during online scan. See page 21 for the syntax description.

Example

```
// TLBL IN01,'SQLAF_2002_04',,VOL001,,,,,DISP=OLD
// TLBL IN02,'SQLAF_2002_05',,VOL002,,,,,DISP=OLD
// EXEC SQLAFATS,SIZE=AUTO,PARM='TAPES=2'
DATABASE SQLDS
TABCREA SQLDBA
TABNAME CUSTOMERS
COLUMN BALANCE < 0
/*
```

The above example will list all customers whose balance became negative during April and May 2002.

7 Archiving the Audit Log

The archive procedure must be scheduled explicitly by the customer, by invoking the SQLAFARC program with following values in the EXEC PARM field:

INTAPE=NO
The SQLAF AUDIT1 file is transferred from disk to the tape named in the TLBL OUT statement. If the TLBL DISP=MOD operand is specified, the AUDIT1 file is appended to the OUT tape. (Examples 1 and 2).

INTAPE=YES
The IN tape is copied to the OUT tape and the SQLAF AUDIT1 file is appended from disk to the OUT tape. (Example 3).

While SQLAFARC is in progress, the Audit Processor SQLAFP will store incoming auditing requests to the secondary log SQLAF AUDIT2. When archiving completes successfully, the Audit Processor performs following actions:

- erase SQLAF AUDIT1
- copy SQLAF AUDIT2 to SQLAF AUDIT1
- erase SQLAF AUDIT2
- resume writing to SQLAF AUDIT1

Examples

(1) The following JCL creates the first generation of an archive tape.

```
// ASSGN SYS021,590
// MTC REW,590
// TLBL OUT,'SQLAF_200205',,VOL001,,,,,DISP=NEW
// EXEC SQLAFARC,SIZE=AUTO,PARM='INTAPE=NO'
```

(2) The following JCL extends the current archive tape.

```
// ASSGN SYS021,590
// MTC REW,590
// TLBL OUT,'SQLAF_200205',,VOL001,,,,,DISP=MOD
// EXEC SQLAFARC,SIZE=AUTO,PARM='INTAPE=NO'
```

(3) The following JCL creates a new generation of an archive tape.

```
// ASSGN SYS020,590
// MTC REW,590
// ASSGN SYS021,591
// MTC REW,591
// TLBL IN,'SQLAF_200205',,VOL001,,,1,,DISP=OLD
// TLBL OUT,'SQLAFARC_200205',,VOL002,,,2,,DISP=NEW
// EXEC SQLAFARC,SIZE=AUTO,PARM='INTAPE=YES'
```

8 Operational Considerations

8.1 Software Prerequisites

- VM/ESA Version 1 Release 1.0 or later
- DB2/VSE Release 3.3 or later
- SQL/Monitoring Facility ("SQL/MF") Version 5.1 or later[9]

8.2 Monitor Dependencies

The Audit Initiator is invoked by the SQL/Monitoring Facility. Therefore, if the monitor is inactive, no auditing will be performed. For instance, if the PERIOD parameter of the SQL/MF configuration file SQLMF CONFIG defines a monitoring period, auditing will occur during this period only. You may have to change the parameter for the purpose of auditing and data security.

8.3 Audit Processor Startup considerations

The Audit Processor accesses the **SQL/MF log database** to retrieve command texts from the SQLCS_SQL_STMNT table. If the log database is down at that time, SQLAFP cannot continue. Operational schedules should be reviewed to ensure that the log database is always up or started first.

8.4 Automatic Audit Processor restart

The Audit Processor SQLAFP **explicitly** restarts itself at 01:00 a.m.

When an abend occurs in the Audit Processor, a dump is sent to the virtual printer, and an **implicit** restart is performed.

[9]SQL/Monitoring Facility is a program product offered by Software Product Research. If the customer does not wish to order SQL/MF, the SQL/MF "command capturing" facility, needed for SQL/AF, can be ordered as a separate product.

8.5 Long-running update jobs

Long-running batch jobs, which perform a large number of commands against an audited table, may fill up the audit request queue or auditing dataspace, with abnormal termination of that job as a result.

It is possible to request SQL/AF to ignore jobs like these:

- by means of the **MONPERIOD** statement in the SQL/MF monitor's SQLMF CONFIG dataset (the auditor is invoked by the monitor)
- by means of the **PERIOD** statement in the SQLAF RULES file (page 11)
- by means of the **IGNORE** statement in the SQLAF RULES file (page 11)

8.6 SQLCODE -901

To prevent applications from executing non-audited commands, the Audit Initiator forces an SQLCODE -901[10] for all SQL statements executed while the Audit Processor server machine is inactive, unless the **UNRESTRICTED** option has been specified on the SQLCS.CONFIG.AUDITOR statement.

If the audit dataspace gets filled up with requests (because it is too small or because the Audit Processor is no longer processing it), SQLCODE -901 will be presented too.

8.7 Auditing a newly created table

When a new table is created and the table name occurs in the SQLAF RULES dataset, either explicitly or implicitly (via a generic tablename), accesses to the newly created table are **not** audited until the next explicit or automatic restart of the Audit Processor.

8.8 Processing overhead due to auditing

The DB2/VSE database servers subject to auditing will store an audit request in the dataspace:

- for all dynamic SQL statements
- for those SQL statements in the application programs that access a protected table
- FETCH commands are never audited

The only overhead incurred by the database server is the time needed to store the data in the dataspace.

8.9 Audit control files

SQL/AF maintains auditing control information in files which have the audited databasename as a filename and one of following filetypes: TABAUDIT, COLAUDIT, TABLIST, COLLIST, VUELIST and VCDLIST.

[10] **-901**: "The current SQL statement has failed due to a system error.
You may continue use of DB2/VSE."

9 Space Estimates

9.1 Space requirements for the audit log

Assuming you requested **AUDIT ALL** for a table, there will be a log entry:

- for each SELECT, INSERT, DELETE or UPDATE command
- for each explicit or implicit COMMIT or ROLLBACK
- for each CLOSE of a cursor-based SELECT
- there are no audit log entries for cursor FETCH commands

Each audit event stored on the log dataset SQLAF AUDIT1 or SQLAF AUDIT2 requires 129 bytes of disk space plus the text of the audited SQL statement.[11] Assuming a command text length of 390 characters as a (pessimistic) average, a log record requires about 512 bytes, giving the following space estimates:

Unit of space (16K record)	Number of log records held
one block	32
one 3390 cylinder	5760
one Megabyte of disk space	8192

During normal auditing, data is written to the **primary log** dataset (SQLAF AUDIT1). However, if the primary log is being archived, incoming audit requests are written to a **secondary log** (SQLAF AUDIT2), until archiving is complete. The contents of the primary log are then deleted and the secondary log is renamed to SQLAF AUDIT1.

9.2 Space requirements for the audit control files

The Audit Processor and the Audit Initiator share a number of control files containing the names of tables and dependent programs to be audited. These files have an 80-byte record for:

- each table to be audited
- each table column to be audited

Two 3380-cylinders should be sufficient to hold these datasets.

[11] The text of static SQL statements is also kept.

10 Audit logrecord layout

The layout of the logrecord on both the disk log and the log archive tapes is as follows:

10.1 Primary logrecord

```
CHAR(8)      Record timestamp (hardware clock value)
CHAR(1)      Flag = 'P' indicating primary audit record
CHAR(10)     Date of auditing in DB2 format yyyy-mm-dd
CHAR(8)      Time of auditing in DB2 format hh:mm:ss
CHAR(8)      DB2 user name performing the command
CHAR(8)      VSE job name performing the command
CHAR(8)      Database containing the protected table
CHAR(8)      Creator of the protected table accessed
CHAR(18)     Name of the protected table accessed
CHAR(8)      Creator of the package containing the command
CHAR(8)      Name of the package containing the command
SMALLINT     Package section number of the command
SMALLINT     Agent number executing the command
INTEGER      DB2/VSE LUW_ID
INTEGER      SQLCODE
INTEGER      Number of rows processed by the command
             (for SELECT this item is stored in the close logrecord)
CHAR(8)      Command type
             ("SELECT","UPDATE","INSERT","DELETE","COMMIT","ROLLBACK")
CHAR(8)      Terminal_identification
SMALLINT     Command mode (1 if a dynamic, 0 if a static command)
SMALLINT     Length of the command text
VCHAR(8192)  Command text (varying length)
```

A primary logrecord is written for all audited commands.

10.2 “Close” logrecord

```
CHAR(8)          Record timestamp (hardware clock value)
                 (same value as the corresponding primary logrecord)
CHAR(1)          Flag = 'C' indicating close audit record
CHAR(3)          Filler
INTEGER          DB2/VSE LUW_ID
                 (same value as the corresponding primary logrecord)
INTEGER          Number of rows processed by the command
CHAR(8)          Name of the package containing the command
SMALLINT         Package section number of the command
```

A close logrecord is written for cursor-based SELECTs when the cursor is closed. The purpose of the logrecord is to record the number of rows actually fetched. The close logrecord can be related to the corresponding primary logrecord (which is written at the opening of the cursor) by means of the record timestamp or the DB2 LUWid (those items are identical in both record types).

11 Messages

11.1 Audit Processor messages

SQLAFP000 Audit Processor loaded at xxxxxx

Informatory message issued during startup of the audit processor.

SQLAFP001 Syntax error in SQLAF RULES.

A syntax error has been detected when decoding the **SQLAF RULES** dataset. The command in error is shown before this message.

SQLAFP002 SQLCODE ... connecting SQL/MF Log database.
Retrying CONNECT until successful.

An SQL error has occurred when SQLAFP attempts to connect the SQL/MF Log database. This database contains the SQLCS_SQL_STMNTS table which is needed during auditing.

This is a severe error that prevents the auditor from continuing its work. Therefore, the auditor will wait until the log database has been started. This may cause delays in the databases that must send auditing requests.

Ensure that the Log database is started and that the audit processor has the necessary SQL CONNECT privileges.

SQLAFP003 SQLCODE ... accessing SYSCATALOG
SQLAFP004 Database=... Creator=... Tablename=...

An SQL error has occurred when SQLAFP attempts to access the DB2/VSE catalog information for the tables in the **SQLAF RULES** dataset.

Ensure that the audit processor has the necessary SQL privileges.

SQLAFP007 Invocation syntax is in error

The invocation parameters on the EXEC SQLAFP statement are invalid.

SQLAFP008 SQLCODE ... accessing SQLCS_SQL_STMNTS

An SQL error has occurred when SQLAFP attempts to access the **Statements** table of the SQL/Monitoring Facility for the packages dependent on the tables in the **SQLAF RULES** dataset.

SQLAFP009 RC ... building list .....

An internal error has occurred when building the named list. Enlarge the virtual storage of the machine running the SQLAFP program. If the error persists, call for software support.

SQLAFP010 VSE file system RC xxx when writing audit lists to disk

A VSE error has occurred when writing the audit control files to disk. Check that enough VSAM space is available for the SPR.PDS cluster.

SQLAFP012 Auditing initialized for database ....

Informatory message issued when an audited database server has been started or when the Audit Processor is initialized after a restart.

SQLAFP013 Initialization of database .... complete

Informatory message indicating that SQLAFP initialization for the database is complete.

SQLAFP014 Secondary log active at startup.

Informatory message indicating that a secondary audit log was found when starting the audit processor. A previous SQL/AF archive procedure did probably not terminate normally. Request an audit archive by invoking the corresponding function of the **SQLAF** program.

SQLAFP016 SQLAF RULES dataset not found

The dataset containing the auditing rules could not be accessed during initialization of the audit processor.

Ensure that the dataset is created on the SQL/AF product disk and that the audit processor is able to access this disk.

SQLAFP017 Database xxx not in SQLAF RULES

During database startup, SQLAFP finds that auditing has been requested for the database in the SQLMF CONFIG file, but finds no entries for the database in the SQLAF RULES dataset. The auditing request is ignored.

Supply auditing rules for the database.

SQLAFP050 VSE file system RC xxx when writing to primary log

A VSE error has occurred when writing the audit log to disk. Check that enough VSAM space is available for the SPR.PDS cluster.

SQLAFP052 Switching to alternate log dataset

Informatory message issued when archiving has been started.

SQLAFP055 Archive has been requested

Informatory message indicating that an automatic archive is in progress.

SQLAFP056 Switching to primary log dataset

Informatory message indicating that the archive procedure has completed normally. The primary log dataset is used again for auditing.

SQLAFP999 Archive Processor shutdown has been completed.

The SQLAF *shutdown* request has been completed. You should restart the SQLAFP program manually.

11.2 Audit Initiator messages

SQLAFP000 Audit Initiator loaded at xxxxxx

Informatory message issued during startup of the audit initiator.

SQLAFI105 SQLCODE -901 forced for package x section y by SQLAFI

Auditing could not be initiated for the named package section because the Audit Processor was inactive or because the auditing dataspace is full. Execution of that section is therefore cancelled.

SQLAFI300 RC x.y creating auditing dataspace

The auditing dataspace could not be created. For an explanation of the return code, refer to the IBM manual **VSE/ESA System Macros Reference** in the section describing the DSPSERV CREATE macro.

SQLAFI301 RC x after ALESERV ADD

The ALESERV ADD terminated with a returncode, which is explained in the IBM manual **VSE/ESA System Macros Reference** in the section describing the ALESERV ADD macro.

SQLAFD204 Auditing dataspace is full

No more space found in the auditing dataspace. The message is given only once in a database session. The situation will be cleared automatically when the Audit Processor has processed queued audit requests.

Unless UNRESTRICTED auditing was requested on the SQLMF CONFIG AUDITOR statement, the LUW being audited recieves SQLCODE -901 when the dataspace is full.

Action

The auditing dataspace may fill up because the Audit Processor machine is not active. You should restart the Processor: this will clear the condition after a few seconds. If the Processor is active, verify its SQL execution status. If in lock or waiting for an agent structure, you must take the appropriate action.

Otherwise, it may be necessary to enlarge the size of the auditing dataspace.

SQLAFD213 Verifying auditing dataspace for database nnn

Informatory message indicating that the logical queue structure of the auditing dataspace is being verified. Verification is done automatically at the start of the Audit Processor and when the “shutdown audit processor” command is issued.

SQLAFD214 Verification completed without errors

Informatory message issued at the successful termination of dataspace verification.

SQLAFD215 Verification of dataspace fails
SQLAFD216 Dataspace is being reformatted

Errors were found in the queue structures of the auditing dataspace. The queue structures are automatically initialized. All auditing requests stored in the dataspace will be lost.

SQLAFD217 <databasename> audit dataspace nn% full

This warning message is issued when the auditing dataspace for the named database server is 80%, 90% or 95% full.

Action

The message may indicate that the dataspace is too small for the average workload. In this case, the auditing dataspace should be enlarged.

If the message is due to a long-running batch job, you may consider forcing the job or disabling auditing for that DB2 user via the IGNORE statement, described on page 12, to prevent future auditing for that job.

11.3 Audit logscan messages

SQLAFATS001 Invalid TAPES parameter

An invalid value was entered in the PARM field of the EXEC SQLAFATS statement.

SQLAFATS002 Syntax error in selection parms

Invalid archive selection parameters were specified after the EXEC SQLAFATS statement.

SQLAFATS003 n records processed on m

Informatory message specifying the number of records from tape **m**.

SQLAFATS004 RC n appending to $$LIST

If the returncode equals 44, more than 32K entries were selected from the audit log. You should specify more scan selection criteria.

Otherwise, you don't have enough virtual storage probably. 8 Mb is required to work comfortably with the logscan program.

11.4 Archive Processor messages

SQLAFARC001 Starting archive

Informatory message issued during startup of the archive processor.

SQLAFARC002 Archive completed

Informatory message issued during termination of the archive processor.

SQLAFARC003 INTAPE parameter invalid

The INTAPE execution parameter is missing or invalid.

SQLAFARC004 n records copied from IN to OUT archive

Informatory message when the INTAPE=YES execution parameter has been specified.

SQLAFARC005 n records added to OUT archive

Informatory message specifying the number of records appended from the audit log to the archive tape.

SQLAFARC006 Waiting to start archive

Archiving cannot be started immediately because the audit processor SQLAFP is processing the audit log.

SQLAFARC007 Execution parameters invalid

An invalid PARM value was entered on the EXEC SQLAFARC statement.