SQL Auditing Facility

User's Guide

Software
Product
Research

SQL/Auditing Facility
Version 2.2

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/VM tables in databases managed by a DB2/VM 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 virtual machine of the database server. Therefore, it is capable of auditing VM/ESA, VSE/ESA and CICS clients, as well as non-DB2/VM clients accessing the DB2/VM 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 a dedicated virtual machine. The Audit Processor writes to the audit datasets, initiates archiving and so on. The Initiator and the Processor components exchange data using IBM's IUCV communications protocol or using SQL/AF Dataspaces.

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 commands accessing the table. Alternatively, one or more table-columns can be defined in the **RULES** dataset: auditing then occurs only when an SQL command refers to one of these protected table columns.

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

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

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

Following transformations are performed on the text of the command

- the command text is shown as it is actually executed by DB2/VM, that is, with all host variable references replaced by the **contents** of the variables[3]
- if the command 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 command does not use a column-list, a default list with all table column names is generated
- INSERT format-2 commands are replaced by INSERT format-1[4]
- cursor-based PUT commands are replaced with an INSERT
- UPDATE or DELETE commands 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

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

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

[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 command for which the Auditor accesses the DB2/VM log.

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

- SELECT commands 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 commands on a protected table are audited. Protected columns are not considered during delete auditing.
- UPDATE commands are audited when one or more protected columns appear in the UPDATE SET or WHERE clauses.
- INSERT or PUT commands are audited when one or more protected columns are inserted by the command.
- 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 log datasets are variable-length CMS files. The primary audit log dataset is always written to the 191 disk (or SFS directory) of the Audit Processor. The secondary log should be placed on a different minidisk.

VM/ESA Version 2 **Data Compression** may be enabled for the audit log, as described on page 79.

[5] FETCH commands 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.

Depending on the expected number of audit events, the user defines an **archive cycle** in the **SQLAF OPTIONS** dataset (see page 31). The archive cycle determines how long the archive tape remains **current**: one day, one week, one month or one year. When archiving is done within the current archive cycle, the disk log is **appended** to the current archive tape. When a new archive cycle begins, the disk log records are stored at the begin of the tape. Moreover, the Archive Processor uses the archive cycle to build the label for the current audit archive tape.

Archiving is usually scheduled explicitly by the user. It may also occur automatically when the primary audit log is full or when a defined number of audit requests have been stored on the primary log.

The archiving function is executed by a service machine running the SQL/AF archive program.

1.4 SQL/AF User Interface

The SQLAF User Interface program allows authorized users to perform the following functions:

- Inspect the primary audit log
- Inspect an audit archive tape
- Initiate an audit log archive
- Shutdown the Audit Processor
- Suspend auditing for a designated database
- Resume auditing for a designated database
- Modify the SQLAF RULES dataset
- Re-apply the GRANT commands in the SQLAF RULES dataset
- Inspect dataspace counters

1.5 Inspecting the Audit Log

A part of the SQL/AF user interface, the ***log scan*** utility program is provided to interactively search the primary audit log or an audit archive tape 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 command text. This powerful scan method examines the text of the SQL commands found in the audit log and selects log entries:
  - where the command text references a named table column
  - where the command 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

1.6 Section Authorities

When requested in the SQLAF RULES dataset, the auditor will implement a new security concept, called **Section Authorities**.

DB2/VM provides **table authorities** (mainly used during dynamic access) and **package authorities**, used to grant access to programs. When access to a package has been granted to a user, that user is able to execute all SQL commands contained in the package. For multi-functional programs (containing both browse and update functions for instance), the **section authorities** concept allows to grant privileges on the individual commands ("sections") of the program. When the SQL/AF **GRANT** command is provided in the SQLAF RULES for a given table, users will not only require the RUN privilege on the program, but also **table authorities** to execute package sections that perform an INSERT, UPDATE or DELETE on that table.

The SQLAF RULES command **GRANT USING SYSTABAUTH** uses the table authorities from the SYSTABAUTH catalog as a base for the section authorities. If this is not desired, because the user should not have dynamic access to that table, DB2/VM-like GRANT commands can be included in the SQLAF RULES dataset.

1.7 Customizing SQL/AF

When provided by the installation, a user exit will be invoked by the Audit Processor for every audit log record written. The exit is written as a REXX program, named **SQLAFUXP EXEC**. On entry, all fields of the logrecord are presented to the exit in the form of REXX variables.

SQL/AF also offers a skeleton for a **User Message Processing** facility that allows applications to send audit-related messages to a central message processor. Both facilities are described on page 67.

1.8 Audit request queueing

Since the auditing process involves at least two virtual machines (an Audit Initiator and the Audit Processor), a queueing mechanism is required to pass the auditing requests from the Initiator to the Processor. By default, an **IUCV** based communication mechanism is used, with queueing provided by the Processor.

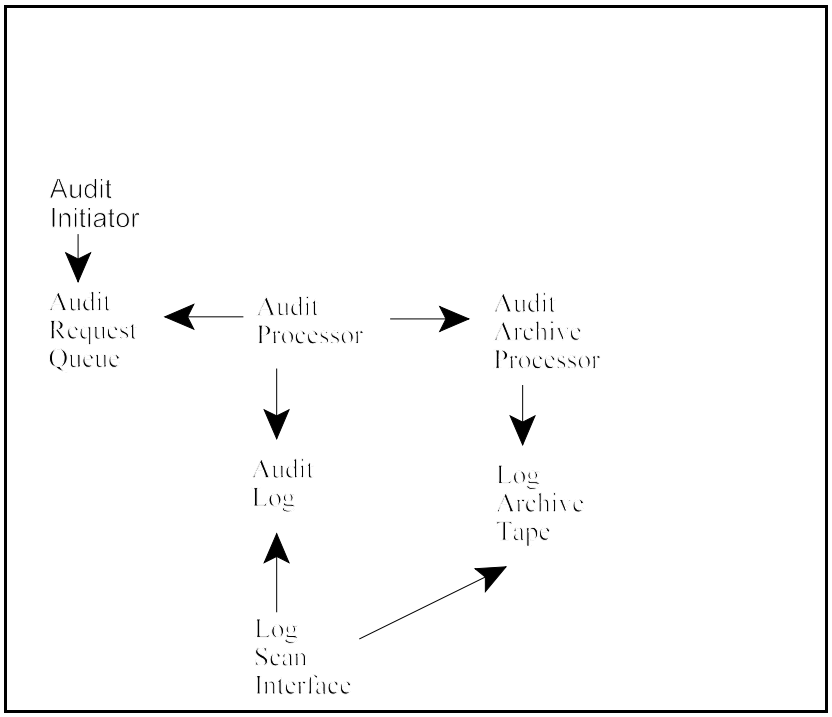
However, if your hardware allows it, you should request SQL/AF to use **dataspaces** as a physical support for the audit request queue. This will greatly improve performance, compared with the IUCV-based queueing technique. If dataspace support has been enabled, the Audit Processor will create a “mapped” dataspace for each database audited. Dataspace mapping (keeping a copy on disk of the dataspace pages in storage) is performed by the VM/ESA Control Program, asynchronously with processing in the Auditor. Moreover, to recover from an occasional dataspace full condition during intensive auditing, a **spill file** may be defined to temporarily store the audit requests not fitting into the dataspace. Using the spill file results in a performance degradation during auditing. Therefore, its use should remain exceptional. Setting up the spill file is described on page 17.

1.9 Command text caching

Before storing the audit log record for a static (compiled) command, the text of the command is retrieved by the Audit Processor from the monitor table SQLCS_SQL_STMNTS. To avoid continuous SELECTs on this table, all command 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 commands that have not been used during a specified period of time.

1.10 Component Overview

- The Audit Initiator runs in the virtual machine of the DB2/VM database server.



- All other components run in their own virtual machine.
- The audit request queue is implemented using dataspaces (the preferred method) or using IUCV communications.
- The audit log is a CMS dataset on the 191 disk of the Audit Processor.

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

1.11 Enhancements implemented by SQL/AF Version 2.2

1. The dependencies between package sections and audited tables are now dynamically analyzed by the Audit Initiator, using information provided by the SQL/Monitoring Facility Version 5.1. Previously, these dependencies were analyzed statically by the Audit Processor at its startup.

2. The Initiator forwards dynamic commands (ISQL, DBSU ...) to the Processor, only when the command makes a reference to an audited table. Previously, all dynamic commands were sent to the Processor for analysis and eventual auditing.

3. The SQL/AF dataspace support introduces the concept of a **spill** file, which allows the dataspace to temporarily overflow into a file when it is full.

4. The SQLAF RULES **IGNORE** statement now has a more flexible syntax and allows to bypass auditing, based on the VM-name, the DB2-name, the package-name or a combination of them. The command is now processed by the Initiator. Previously, it was handled by the Processor and the Initiator still performed request queueing for ignored SQL commands.

5. The SQLAF RULES **PERIOD** statement is now processed by the Initiator. Previously it was handled by the Processor and the Initiator still performed request queueing for SQL commands executed outside the auditing period

6. When obtaining the text for audited commands, the Audit Processor now keeps the text of the most frequently used commands in lookaside buffers. This will reduce the number of accesses to the SQL/MF SQL_STMNTS table.

2 Installing the Audit Facility

2.1 Installing the CMS components

This installation step loads the SQL/AF modules, execs, help files and other material from the distribution tape to the disk or directory where the SQL/Auditing Facility must be installed (the ***SQL/AF product disk***). Since the Auditing Facility is an extension of the SQL/Monitoring Facility, the SQL/MF product disk must also be accessed during SQL/AF operation. For simplicity of operations, you may decide to place the SQL/AF material on the SQL/MF product disk.

2.1.1 Prerequisites

1. Ensure that the VM userid performing the installation has write access to the SQL/AF product disk.
2. Ensure that read access has been established to the SQL/MF product disk.
3. Ensure that you have access to the DB2 production minidisk with any filemode.
4. Ensure that a tape or cassette device has been attached to the virtual machine with virtual address 181.
5. The SQL/AF software requires 10 additional 3380-cylinders on the SQL/AF product disk.

2.1.2 Tape Installation

1. Access the product disk for write with CMS filemode A.
2. Issue the command: **TAPE LOAD * * A**
3. When the TAPE LOAD command has completed, you do no longer need the distribution tape.

2.1.3 E-mail Installation

If you received the SQL/AF product by electronic mail, follow the installation instructions provided in the README file.

2.1.4 Linking the SQL/AF Modules

1. With the product disk still accessed as "A" enter **SQLAFPI** at the CMS prompt.
2. On the installation option screen, enter **Y** on the line **CMS Installation (Y/N)**.
3. The SQL/AF modules are linked on the product disk.

2.2 Installing the DB2 components

The DB2 related installation must be performed for each database where the auditing facility will be used.

2.2.1 Prerequisites

1. You should have write access to the SQL/AF product disk in any filemode.
2. You should have read access to the SQL/MF product disk in any filemode.
3. You should have read access to the DB2 production minidisk in any filemode.
4. You should be able to specify the password of DB2 user **SQLDBA**.

2.2.2 Installing

1. Type **SQLAFPI** at the CMS prompt.
2. On the installation option screen, enter **Y** on the line **SQL Installation (Y/N)**.
3. On the next panel, fill in the following fields:

   - the name of the database for installation
   - the password of user SQLDBA in this database
   - the name of the virtual machine running the Audit Processor as defined in the section **Setup the Audit Processor** on page 18.

4. Install will load the SQL/AF packages in the database and grant DBA authority to the default userid of the Audit Processor.[7]

5. The previous panel is shown again, to allow for installation in another database. Press PF3 to terminate the installation procedure.

DB2 installation must be performed:

- in the “Log” database of the SQL/Monitoring Facility
- in all databases that will invoke the Auditing Facility.

[7]DBA authority is required because the Audit Processor may need to unload audited packages. GRANT DBA is done for the VMID of the Audit Processor *without a password*, so that the SQLID can be used by the Audit Processor only.

3 Setting up the Audit Facility

3.1 Setup the IUCV communications environment

IUCV communications are used by the SQL/Auditing Facility:

- to pass the auditing requests if SQL/AF dataspace support is not used
- to pass occasional control signals between the Audit Initiator and the Audit Processor

If SQL/AF dataspace support is used, auditing requests are passed by the Initiator to the Processor by means of the dataspace, not by IUCV. IUCV is used as the communications vehicle when dataspace support has not been enabled. Without dataspaces, intensive IUCV communications occur. With dataspaces, IUCV usage is limited to the passing of a synchronization signal, when there has been no auditing during 30 seconds.

To allow the Initiators in all audited databases to communicate with the Audit Processor, insert an IUCV ALLOW statement in the directory entry of the Audit Processor.

Define the Audit Processor machine as a DB2 client in all audited databases and in the database that contains the tables of the SQL/MF Monitoring Facility. This will also enable the Audit Processor to send IUCV messages to the Initiators.

3.2 Setup the audit queue in dataspaces

If your installation does not have support for dataspaces, skip the following sections and continue with ***Setup the Auditing files*** on page 16.

3.2.1 Enable Dataspace Support in the CP directory

- Ensure that all database servers and the virtual machine running the SQLAFP program are defined as XC machines in the VM/ESA directory

- In the directory entry of the machine running the Audit Processor, include the following directory control statements:

  - *XCONFIG ACCESSLIST ALSIZE* ***a***, where **a** equals the total number of databases audited

  - *XCONFIG ADDRSPACE MAXNUMBER* ***a*** *TOTSIZE* ***b****M SHARE*, where **a** equals the total number of databases audited and **b** equals the total size in mega- or gigabytes of the dataspaces for the above databases. The size of a dataspace depends on the size of its underlying MAPDISK, as will be explained later.

- In the directory entry of all audited database servers, verify that the ALSIZE operand of the XCONFIG ACCESSLIST directory statement allows for the additional ALET needed by the Audit Initiator to address the auditing dataspace.

3.2.2 Preparing the dataspace mapping disks

Since SQL/AF uses **mapped** dataspaces[8] a number of mapping disks must be prepared.

Each audited database needs at least **one** mapping disk. Several mapping disks can be specified per database thus distributing the entire mapping area over several minidisks. Multiple mapping disks per database can be used for I/O performance reasons or if a sufficiently large contiguous extent is not available. It also allows to extend the dataspace more easily later on, by adding a new mapping minidisk.

The mapping disks are prepared as follows:

- In the directory entry of the Audit Processor, define as many minidisks as there are audited databases (or more if multiple mapdisks are used per database). Note that the size of the auditing dataspace created depends on the size of its mapping disk (in 4K pages). The space requirements for a dataspace are discussed in the chapter ***Space Estimates*** on page 77.
- Format each mapping minidisk using 4K blocks. You **must** specify a disk label during formatting.
- Access each minidisk as <fm>
- Issue the command **RESERVE SQLAF MAPDISK <fm>**.
- Insert the statement **MAPDISK <database_servername> <device_address>** for each mapping disk in the **SQLAF OPTIONS** dataset.
- The total number of MAPDISK statements in SQLAF OPTIONS should not exceed 256.
- The Audit Processor dynamically accesses the mapping disks when needed during execution: do not issue ACCESS commands for them in the PROFILE.

[8]The contents of a mapped dataspace are automatically and asynchronously copied to a *mapping disk* by the VM control program.

3.2.3 Map disk setup example

DBN1 and DBN2 are audited database servers. DBN1 has two mapping disks, DBN2 has one. The size of the auditing dataspace for DBN1 is 7200 pages, and 3600 pages for DBN2. Taking 1K as the average length of a queued audit request, 28800 requests could be queued for DBN1, 14400 for DBN2.

Update the Directory for the Audit Processor

- MDISK 400 3390 10 20 VOLXXX W passwords
- MDISK 401 3390 100 20 VOLXXX W passwords
- MDISK 402 3390 150 20 VOLXXX W passwords

Prepare the mapping disks (in the Audit Processor machine)

- FORMAT 400 C (BLKSIZE 4K
- RESERVE SQLAF MAPDISK C
- FORMAT 401 C (BLKSIZE 4K
- RESERVE SQLAF MAPDISK C
- FORMAT 402 C (BLKSIZE 4K
- RESERVE SQLAF MAPDISK C

Update SQLAF OPTIONS

- MAPDISK DBN1 400
- MAPDISK DBN1 401
- MAPDISK DBN2 402

3.3 Setup the auditing files

3.3.1 Description of the auditing files

During auditing following CMS files are written:

1. The primary audit log dataset SQLAF AUDIT1. It contains the audit output.
2. The secondary audit log dataset SQLAF AUDIT2. It 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.
4. The optional audit spill file. It is used only with SQL/AF dataspace support.

The spill file is written by the Audit Initiator on the spill minidisk, if one has been defined. All other files are written by the Audit Processor, in filemode A (except for the secondary log which is normally written to another disk or directory).

To determine the size of the above files, please refer to ***SQL/AF Space Estimates*** on page 77. Most of the allocated space however will be taken by the primary log dataset.

3.3.2 Placement of the audit datasets

Audit datasets can be placed on CMS minidisks (“EDF format”) or in SFS directories. The following setup is suggested:

- datasets to be placed on minidisk
  - the primary audit log dataset SQLAF AUDIT1
  - the audit control datasets
- datasets to be placed on SFS (to benefit from the space allocation flexibility provided by SFS)
  - the secondary audit log dataset SQLAF AUDIT2
  - the optional audit spill file (only when SQL/AF dataspace support is used)

3.3.3 Setup the spill file

The spill file is used only when dataspace support has been enabled. Its purpose is to handle occasional dataspace overflow situations at moments of intensive auditing (caused by long running update jobs for example). To recover from a dataspace full condition, the Audit Initiator uses the spill file to temporarily store the overflowing audit requests on the file and to store them back into the dataspace when the dataspace full condition has been cleared. The spill file is not used (and does not exist), when all auditing requests can be passed through the dataspace.

Because of its dynamic nature and its highly variable size, the spill file is best defined in an SFS directory. Using the spill file results in a performance degradation during auditing. Therefore, its use should remain exceptional.

The CMS filemode for the spill file directory or minidisk is specified on the **SPILL_FM** statement in the **SQLAF OPTIONS** dataset. The spill directory or minidisk must be accessed in the DB2/VM startup stream, before the EXEC SQLCSI statement.

3.3.4 Granting file access privileges

Following SQL/AF components need minidisk read access or the SFS read privilege on **all files** on the A-disk or directory of the Audit Processor:

- all database servers being audited
- the archive processor
- the VMID's that will invoke the SQL/AF User Interface

3.3.5 Additional file setup considerations

- If the primary log is in SFS, define its directory name in the **SQLAF OPTIONS** dataset (see page 31).

- If the primary log is placed in SFS, the secondary log may be specified in the same directory of the primary log, if that filepool has sufficient space to contain both.

- If the primary log is not placed in SFS, an SFS directory should be defined for the secondary log. Although the secondary log can be placed on the A-disk, this would cause abnormal termination when auditing requests arrive during an implicit archive due to a "primary log full" condition. The space required for the secondary log depends on the estimated number of audit requests that will be received while the audit archive procedure is in progress.

3.4 Setup the Audit Processor

Create a virtual machine with a name of your choice. It will run the **Audit Processor** function (the **SQLAFP** program).

The virtual storage for this machine should be specified as 16 MB at least. We recommend a virtual storage size of 32 Mb. If a large number of packages is involved in auditing, more storage may be required to implement the command cache. For each static (compiled) SQL statement audited, a cache entry is created with a length equal to 50 + the length of the command_text.[9] The CLEANUP argument specified during SQLAFP startup may request periodical clearing of the command cache (see page 34).

The directory entry for the server should contain the IUCV ALLOW statement.[10] (Even when SQL/AF Dataspace support is used, IUCV communications are still needed to pass occasional control messages and commands). It is suggested that the VM single console facility be enabled for this machine.

In the PROFILE of the Audit Processor, insert the CMS commands necessary to access

- the SQL/MF product disk
- the SQL/AF product disk
- the DB2/VM production minidisk
- the secondary audit log (if appropriate)

Then invoke the SQLAFP program as described in *Enabling the Audit Processor* on page 34.

Insert the necessary auto-logon commands for the machine in the VM startup stream. In addition, if the **AUDITOR** facility of IBM's *CMS Utilities* Program Product has been installed on your system, you should insert the machine as a candidate for automatic logon.

[9]The text before hostvariable substitution, as found in the monitor table SQLCS_SQL_STMNTS.

[10]There is no security exposure in allowing global IUCV connect, since the server does not process messages other than those generated by other SQL/AF components.

3.5 Setup the Audit Archive Processor

Create a virtual machine that will implement the **Audit Archiving** function (the **SQLAFA** program). The default name of this machine is **SQLAFARC**. If you define your own name, you must specify it when starting SQLAFP (see *Enabling the Audit Processor* on page 34).

The virtual storage for this machine should be specified as 4 MB at least.

The directory entry for this server should contain the IUCV ALLOW statement. It is suggested that the directory also contains the statement **XAUTOLOG <name-of-audit-processor>**. This enables the Audit Processor to issue an XAUTOLOG command and continue execution, when it finds that the Archive Processor is not logged on.

If your installation does not use tape managing software, the archive processor displays archive tape mount messages on its console. The VM single console facility should be enabled for this machine, to direct tape mount messages to operations.

The archive function does not require A-disk space beyond the CMS minimum, except when archiving a compressed log. In this case, decompression disk space is needed, as described on page 80.

The function does not access DB2 databases.

In the PROFILE of the machine, insert the commands necessary to access the SQL/MF and SQL/AF product disks and initiate audit archiving by inserting the command **EXEC SQLAFA**. For more details, see *Enabling SQL/AF* on page 19.

Insert the necessary auto-logon commands for the machine in the VM startup stream. In addition, if the **AUDITOR** facility of IBM's *CMS Utilities* Program Product has been installed on your system, you should insert the machine as a candidate for automatic logon.

3.6 Tape User Exit

When your installation uses tape managing software, the SQL/AF Archive Processor and the LogScan program may be requested to call the user exit **SQLAFUTX**, for tape mounting and dismounting control.

A dummy SQLAFUTX is distributed with filetype **EXECS**. To activate the exit, change its filetype to **EXEC** and insert your coding in the EXEC as described in the following section.

Please note that following sample EXECs are also distributed with SQL/AF:

- SQLAFUTX DYNAMT: sample exit for use with DYNAM/T of Computer Associates
- SQLAFUTX VMTAPE: sample exit for use with VMTAPE of Sterling Software

If you are using one of these tape managers, you may find the sample useful and modify it to suit the requirements of your installation.

At the start of the archive, SQLAFUTX is called with function code **Q_EXTEND** to determine whether your tape manager allows tape extend via the FILEDEF DISP MOD operand. (Tape extend occurs when audit log entries must be written to the archive tape of the current archive period.) If your tape manager allows tape extend, code the statement **returncode = 1** in the **query_extend** routine of SQLAFUTX. If not, leave **returncode = 0** as set at the beginning of SQLAFUTX.

3.6.1 User exit calls during archive (extend enabled)

When you did enable tape extend, following SQLAFUTX calls will be made during archive:

SQLAFUTX CREATE <tape_label>

The CREATE function is called when a tape must be allocated for a new archive period. The exit should request allocation of a scratch tape labelled <*tape-label*>. The tape must be mounted on TAP1.

SQLAFUTX OPEN <tape_label>

The OPEN function is called when the tape for the current archive period must be mounted. The exit should request mounting of the tape labelled <*tape-label*> as device TAP1.

SQLAFUTX CLOSE <tape_label>

The CLOSE function is called at the end of the archive. The exit should request dismounting of the tape labelled <*tape-label*>.

3.6.2 User exit calls during archive (extend disabled)

When tape extend has been disabled, SQLAFUTX will subsequently be called as follows:

SQLAFUTX CREATE <tape_label>

The CREATE function is called when a tape must be allocated for a new archive period. The exit should request allocation of a scratch tape labelled <*tape-label*>. The tape must be mounted on TAP1.

SQLAFUTX OPEN <tape_label>
SQLAFUTX OPEN_AUX <tape_label>

The OPEN function is called when the tape for the current archive period must be mounted. The exit should request mounting of the tape labelled <*tape-label*> as device TAP1. The OPEN_AUX function requests allocation of a second tape as device TAP2. The archive processor will copy the contents of the primary tape (TAP1) to the auxiliary tape (TAP2) and append the disk audit log to the auxiliary tape. Typically, the primary and secondary tape files will be members of a generation group.

SQLAFUTX CLOSE <tape_label>
SQLAFUTX CLOSE_AUX <tape_label>

The CLOSE and CLOSE_AUX functions are called at the end of the archive. The exit should request dismounting of the primary and auxiliary tapes labelled <*tape-label*>.

3.6.3 User exit calls during Logscan

The Logscan utility can be requested to process an audit archive tape. If the SQLAFUTX EXEC is found, the exit will be called during logscan as follows:

SQLAFUTX OPEN <tape_label>

The OPEN function is called at the begin of Logscan to mount the tape label specified by the logscan user. The exit should request mounting of the tape labelled <*tape-label*> as device TAP1.

SQLAFUTX CLOSE <tape_label>

The CLOSE function is called at the end of the logscan. The exit should request dismounting of the tape labelled <*tape-label*>.

If the tape exit executes satisfactorily for SQL/AF archiving, it will also work correctly during LogScan.

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 CMS dataset to be setup by the user. The dataset should be located on the SQL/AF product disk.

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

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

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.
6. The **GRANT** statement may be specified as often as needed to define the users authorized on the table.

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 [auto-]restarted.

However:

- the SQLAFCMD NEWRULES command can be used to re-apply the RULES without restarting the Audit Processor
- when GRANT commands only have been modified, a user interface function is provided to apply the grants without restarting the Audit Processor. This user interface function is described on page 52.

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 **SQLCS CONFIG** dataset is in effect, auditing will be suspended during the monitor's inactivity.

4.1.2 Specifying audit ignore rules

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

Designates the VMIDs, 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 (VMID, SQLID, Packagename).

If one of the keywords VMID, 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 VMID dbamaint SQLID sqldba PACKAGE sqldba.aridsql
Disables auditing for SQLDBSU when executed under the specified VMID and SQLID.

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

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

4.1.3 Specifying the audited database

Syntax : DATABASE name **VMID** database_server_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. The **VMID** clause designates the virtual machine name of the database server. 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.[11] The COLUMN and AUDIT statements following a generic TABLE statement, will apply to all the tables satisfying the generic name.[12]

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.

[11]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.

[12]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 commands 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 Defining section authorities using SYSTABAUTH

Syntax : GRANT USING SYSTABAUTH

Requests that section authorities should be enforced for all packages performing DELETE, INSERT or UPDATE on a given table. The names of the authorized users are obtained from the SYSTABAUTH catalog rows for this table.

Section authorities are not checked for users with DBA authority. Only those table operations for which auditing has been requested in the AUDIT command are subjected to section authorities (AUDIT CHANGE or AUDIT ALL is required to implement section authorities for insert, update and delete).

Violation of section authorities results in SQLCODE -551. The field SQLERRP will be set to “SQLAFI” and the SQLERRM field will contain the username, the illegal action attempted and the name of the table.

When GRANT commands have been modified, a user interface function is provided to apply the grants without restarting the Audit Processor. This user interface function is described on page 52.

The GRANT command should be specified after the TABLE command to which it applies.

The **Section Authorities** concept is described on page 5.

4.1.8 Explicitly defining section authorities

Syntax : **GRANT{ DELETE | INSERT | UPDATE | ALL } TO username**

Indicates that section authorities should be enforced for all packages performing DELETE, INSERT or UPDATE on a given table and that no table authorities are available for **username** in SYSTABAUTH. Instead the DELETE, INSERT or UPDATE authority is granted only for checking the section authorities. A GRANT command cannot specify more than one authority. If multiple authorities are to be granted to the same user, each one must be coded on a new GRANT command.

The GRANT ALL specification is a shorthand for GRANT DELETE, INSERT and UPDATE.

Specific GRANTs may be combined with the GRANT USING SYSTABAUTH command, in which case section authorities will be granted to the users in SYSTABAUTH and to those named in specific GRANTs. Duplicate GRANTs are recognized by the auditor: all grant specifications for a given user on a given table are compressed into a single internal authorization entry.

Only those tables for which auditing has been requested in the AUDIT command are subjected to section authorities.

Violation of section authorities results in SQLCODE -551. The field SQLERRP will be set to “SQLAFI” and the SQLERRM field will contain the username, the illegal action attempted and the name of the table.

When GRANT commands have been modified, a user interface function is provided to apply the grants without restarting the Audit Processor. This user interface function is described on page 52.

The GRANT commands should be specified after the TABLE command to which they apply.

The **Section Authorities** concept is described on page 5.

4.1.9 Comments

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

4.1.10 Sample SQLAF RULES

```
    DATABASE databasename VMID servername
*
    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
*
    TABLE SQLDBA.T4   (5)
    AUDIT ALL
    GRANT ALL TO USER1
    GRANT INSERT TO USER2
```

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. Application of the section authorities is requested for the table SQLDBA.T4. The related GRANT commands state that USER1 is able to execute package sections that insert into, delete from or update the table. USER2 is allowed to insert into the table. Other users will receive SQLCODE -551 if they execute a package section that modifies the table, even if they have the package RUN authority.

4.2 Defining auditing options

Additional processing options for the SQL/AF program product can be defined in the CMS file **SQLAF OPTIONS**. This optional dataset should be located on the SQL/AF product disk. If the dataset does not exist, default values will apply. The SQLAF OPTIONS dataset may contain following statements:

4.2.1 Specifying an archive cycle

Syntax : ARCHIVE_CYCLE {DAY | WEEK | MONTH | YEAR}

During the archive procedure, the primary audit log is appended from disk to the current archive tape. The archive_cycle command allows you to specify whether you will keep an archive tape per day, per week, per month or per year. If you omit the statement, ARCHIVE_CYLE will be set to **MONTH**.

The archive processor uses this specification
- to detect the begin of a new archiving cycle
- to request mounting of a new or an existing archive tape
- to create a new archive or to append to an existing archive tape
- to build the archive tape label

For more details please refer to page 63.

4.2.2 Specifying the SFS directory of the audit log

Syntax : AUDIT_DIR <directory-name>

This statement should be specified only when the audit log dataset is located in an SFS directory. Since the Audit Processor writes the log and its other control files using filemode A, the entire A-disk should be placed in that directory. The AUDIT_DIR statement is provided to allow external components, such as the User Interface, to access this directory. (This is done by the SQLAFLNK EXEC, using the CMS command **ACCESS directory-name filemode**). Therefore **directory-name** should specify the complete directory name, including the name of the filepool. If AUDIT_DIR is not specified, SQL/AF assumes that the audit log is placed on the A-minidisk of the Audit Processor.

4.2.3 Defining a dataspace mapping disk

Syntax : MAPDISK <database_servername> <device_address>

This statement enables the SQL/AF dataspace support facility.

database_servername
specify the name of the DB2/VM database server that will store auditing requests into the dataspace corresponding to the mapdisk

device_address
specify the device address of this mapdisk, as defined in the MDISK directory control statement for the Audit Processor machine.

For more details on SQL/AF dataspace support, refer to page 13.

4.2.4 Specifying the spill filemode

Syntax : SPILL_FM <database> <filemode>

If you are using SQL/AF dataspace support, code this statement to enable spill file support. Enter the databasename and the CMS filemode that has been used to access the spill file in that database server. If the spill file must reside in SFS, you should have created the spill file directory. If a minidisk is used, it must have been prepared. In both cases, a CMS ACCESS must be performed in the DB2 startup stream, using the filemode specified on the SPILL_FM statement.

For further information, see “**Setup the spill file**” on page 17.

4.2.5 Requesting explicit dataspace save

Syntax : DSPSAVE

When the DSPSAVE option is not specified, SQL/AF entirely relies on VM/ESA to ensure that the dataspace storage pages are saved (“mapped”) to disk. However, hardware failures, such as power failures, may cause uncontrolled termination of VM/ESA. In these cases, some data in the auditing dataspace could be lost. With the DSPSAVE option, SQL/AF explicitly requests VM/ESA to save the modified audit dataspace pages at the end of each LUW. The DSPSAVE option however will cause a certain performance degradation.

5 Starting the auditor components

5.1 Starting the Audit Initiator

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

The AUDITOR statement is used to communicate the name of the virtual machine running the Audit Processor (the **SQLAFP** program) to the SQL/Monitoring Facility.

Its syntax is as follows:

AUDITOR audit_processor_name [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/VM 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 SQLCS 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.

Prerequisites

- access must have been established to the SQL/MF product disk
- access must have been established to the SQL/AF product disk (if different from the SQL/MF product disk)

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 [ARCHLIM n]
[ARCHPROC xxxxxxxx]
[CLEANUP n]
[DSPOLL n]
[RESTART hh:mm]
[SLOGFM x]
[SQLDSF]

ARCHLIM

Specify the archive threshold as the number of records to be written on the primary audit log (SQLAF AUDIT1) before automatic archiving occurs. If omitted, automatic archive occurs only when the primary log is full.

ARCHPROC

Specify the name of the virtual machine that runs the Archive Processor program **SQLAFA**. If omitted, ARCHPROC defaults to **SQLAFARC**.

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 commands remains in storage during the entire Audit Processor session. CLEANUP should be requested only when not enough storage is available to retain all audited commands in the cache.

DSPOLL

If SQL/AF dataspace support has been enabled (by including MAPDISK statements in the SQLAF OPTIONS dataset), this argument specifies the time interval used by the Audit Processor to inspect the dataspaces for auditing requests. This is called the *primary polling interval*.[13] It is expressed in seconds and defaults to 1 second. Specifying large polling intervals usually requires a larger auditing dataspace because requests stay in the audit queue for a longer period of time.

[13]When no auditing requests are received during a number of consecutive pollings, a *secondary* polling interval of (DSPOLL * 3) is activated. The arrival of an audit request reinstates the primary polling interval.

RESTART

Designates the time at which SQLAFP will start a new auditing session. SQLAFP auto-restart performs a CMS IPL and an execution of the PROFILE EXEC. If omitted, restart occurs at 01:00 am.

SLOGFM

Specify the CMS filemode of the minidisk or directory containing the secondary audit log dataset. You must have accessed this disk or directory prior to calling SQLAFP. If SLOGFM is omitted, the secondary filemode defaults to A.

SQLDSF

Code the **SQLDSF** keyword if the SQL/DataSync facility has been installed. This ensures that the audit records written by SQLAFP can be used as input for the DataSync facility.

Prerequisites

- access must have been established to the SQL/MF product disk
- access must have been established to the SQL/AF product disk (if different from the SQL/MF product disk)

Notes

- The Audit Processor performs SQL access during operation, to retrieve the command text for audited package sections from the SQLCS_SQL_STMNTS table, which resides in the SQL/MF “Log” database. Enough agent structures should be provided in that database to accommodate these accesses.

- All audit datasets, except the secondary log, are accessed with filemode A. If the audit directory has not been accessed as the **top directory** at IPL, you must issue the ACCESS command before starting the Audit Processor.

5.3 Starting the Audit Archive Processor

The Audit Archive Processor runs in a dedicated machine and is started by including the command **EXEC SQLAFA** command in the PROFILE EXEC. The SQLAFA program then goes into a non-ending wait for an archive request from the Audit Processor.

Prerequisites

- access must have been established to the SQL/MF product disk
- access must have been established to the SQL/AF product disk (if different from the SQL/MF product disk)

5.4 Initiating the audit archive scan server

The archive scan server allows users to send archive scan parameter files for processing in “batch” mode.

To setup the server, following actions should be taken:

- Define a virtual machine with an A-disk, large enough to contain the largest scan output file.

- Ensure that the server is autologged.

- In the PROFILE EXEC of the machine, insert the command **EXEC SQLAFBTS [hh:mm]** where hh:mm is the time when SQLAFBTS should start processing all the scan parameter files it received in its reader. When a start time is not declared, processing starts as soon as a scan parameter file arrives.

- Unless you are using tape manager software, the server will issue messages to attach a tape drive and mount the archive tape(s). In this case, a secondary console should be defined for the server machine. If a tape manager is used, mounting and dismounting can be controlled using the SQLAFUTX user exit.

6 User Interface

Prerequisites

- access must have been established to the SQL/MF product disk
- access must have been established to the SQL/AF product disk (if different from the SQL/MF product disk)

To invoke the SQL/AF User Interface, enter the command **SQLAF** at the CMS prompt. The following function menu is then displayed:

```
              *--------------------------------------*
                         SQL/Audit Facility
              *--------------------------------------*

                    Scan Current Audit Log
                        Scan Audit Archive
                    Initiate Audit Archive
                  Shutdown Audit Processor
                          Suspend Auditing
                           Resume Auditing
                      Setup Auditing Rules
                           Re-apply GRANTs
                       SQLAF RULES history
                 Disable user for auditing
                  Enable user for auditing
                  Audit dataspace counters

Place the cursor on one of the above functions and press ENTER
```

The SQLAF User Interface program allows you to perform following functions:

- Inspect the primary audit log, using the LogScan program
- Inspect an audit archive tape, using the LogScan program
- Initiate an audit log archive
- Shutdown the Audit Processor (SQLAFP)
- Suspend auditing for a designated database
- Resume auditing for a designated database
- XEDIT the SQLAF RULES dataset
- Re-apply the GRANT commands in the SQLAF RULES
- Disable or enable auditing for a named DB2/VM userid

To select the desired function, place the cursor on the corresponding screen line and press ENTER.

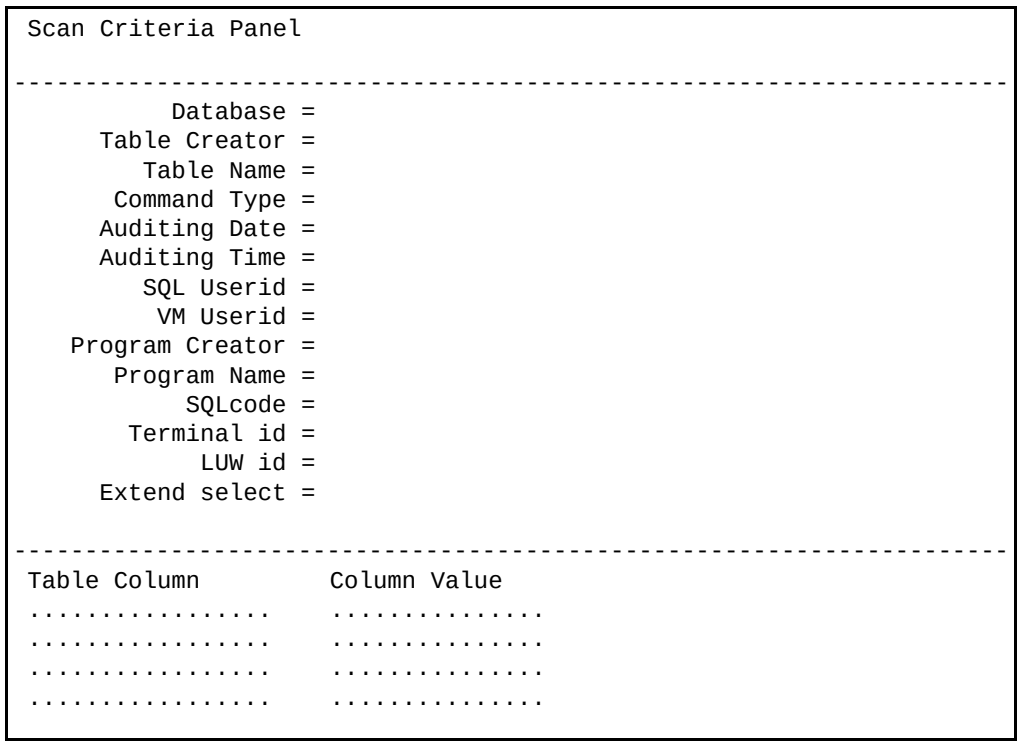
6.1 Scanning the Audit Log

The LogScan program displays selected audit entries from the primary audit log (SQLAF AUDIT1 on the A-disk or directory of the Audit Processor) or from an audit archive tape.

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 commands.

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.
PF2 Recalls the audit log rows obtained during the previous log scan (provided it occurred in the same CMS session).
PF3 Quits the selection panel.
PF4 Saves the currently selected log rows to a CMS file named **AFyymmdd SLhhmmss**. This file can be re-viewed later on by typing **SQLAF** on its filelist line.

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
- VM user name performing the access
- package creator and package name performing the access
- command's SQLCODE as *-nnn*
- user's terminal number
- LUW number

The field **Extend select** can be used to extend the current log scan selection set, resulting from a previous log scan during the current CMS session. The latest log scan selection data are always kept in memory, until the next CMS IPL. When extend select is set to **Y**, the newly selected log rows will be appended to the existing selection set.

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 VM_User  Term
-------------------------------------------------------------------------------
SQLDBA.EMPLOYEE        1997-04-29 11:15:06 SPRDB2   METAST2  METAST2  T25T
SQLDBA.EMPLOYEE        1997-04-29 11:15:52 SPRDB2   METAST2  METAST2  T25T
SQLDBA.EMPLOYEE        1997-04-29 11:16:34 SPRDB2   METAST2  METAST2  T25T
SQLDBA.EMPLOYEE        1997-04-29 11:16:43 SPRDB2   METAST2  METAST2  T25T
SQLDBA.EMPLOYEE        1997-04-29 11:18:36 SPRDB2   METAST2  METAST2  T25T
SQLDBA.EMPLOYEE        1997-04-29 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 command 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 **SQLAFPRT** dataset. The physical destination of the hardcopy dataset is determined by the **SQLAFPFD** EXEC. By default, the hardcopy is directed to the virtual printer. The SQLAFPFD EXEC may be modified to direct the output to another destination, such as a CMS file.

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 46.

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  1997-04-29          Time  11:19:10        Database   SPRDB2
 SQL_User  METAST2           VM_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 Sample Log Scan Session

```
SQL/Auditing Facility                © 1996 Software Product Research
Scan Criteria Panel
--------------------------------------------------------------------------
          Database
     Table Creator
        Table Name    EMPLOYEE
       Command Type
     Auditing Date    1998-04%
     Auditing Time
        SQL Userid
         VM Userid
   Program Creator
      Program Name
           SQLcode
       Terminal id
     Extend select
--------------------------------------------------------------------------
Table Column          Column Value
EMPNO                 100<
```

The sample scan criteria specified on the above panel will show all accesses performed to the employee table in April 1998. It will select all recorded commands for the table, where the command text contains any reference to the column EMPNO with a column value lower than 100.

The output panels will be returned in the following format:

```
SQL/Auditing Facility - Scan Report                          Page 1 of 6
 ------------------------------------------------------------------------
TableName  SQLDBA.EMPLOYEE
AuditDate  1997-04-29  Time     11:15:06   Database   SPRDB2
SQL_User   METAST2     VM_User  METAST2    Terminal   T25TT261
Commit     Y           ProgCrea SQLDBA     ProgName   SQLCLEXP
Section    9           Commtype INSERT     Rows_Proc  1
SQLCODE    0           SQL_LUW  1161151

 ------------------------------------------------------------------------
INSERT INTO SQLDBA.EMPLOYEE ( EMPNO , FIRSTNME , MIDINIT , LASTNAME ,
WORKDEPT, PHONENO , HIREDATE , JOB , EDLEVEL , SEX , BIRTHDATE , SALARY
, BONUS , COMM) VALUES ( '000035' , 'SALLY' , 'A' , 'FIELDS' , 'C01' ,
'4738' , '1998-04-05', 'MANAGER' , 20 , 'F' , '1941-05-11' , 38250.00 ,
800.00 , 3060.00 )
```

```
SQL/Auditing Facility - Scan Report                          Page 6 of 6
 ------------------------------------------------------------------------
TableName  SQLDBA.EMPLOYEE
AuditDate  1997-04-29   Time     11:19:10     Database   SPRDB2
 SQL_User  METAST2      VM_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 WORKDEPT = 'C02' WHERE EMPNO = '000055'
```

6.3 Scanning an Audit Archive tape

First, if the scan program does not find the SQLAFUTX tape exit, it assumes that no tape managing software is available and it will request to manually attach a tape device to your virtual machine. When this has been done, press ENTER to continue.

Next, the program requests the label of the archive tape to process. The *"tape label"* screen field shows the format of the tape label, which depends on the ARCHIVE_CYCLE specification in the SQLAF OPTIONS file. By default, a monthly archive tape cycle is assumed and the tape label will have the format **SQLAF-yyyy-mm**.

The tape label which you entered at the last invocation of the utility will be shown as the default.

Complete the tape label and press ENTER. Processing continues as for a primary audit log scan, as described on page 38.

6.4 Scanning an Audit Archive tape in batch mode

6.4.1 Preparing the scan parameter file

Copy the file **SQLAFBAT COPY** from the SQL/AF product disk to your A-disk as <yourname> **SQLAFBAT A**.

XEDIT <yourname> **SQLAFBAT A** and complete the REXX variable assignments it contains. These REXX variables are used:

- To define the time of execution for your scan as **hh:mm** in the variable **Exec_time**. The execution time should be specified only if you intend to execute the scan in your own machine.
- To specify the label(s) of the archive tape(s) to be scanned in the variables **Filedef.1**, **Filedef.2** etc. There is no limit to the number of tape labels that can be named.
- To specify in the variable **Secuser** the name of the machine to which the tape attach and mount messages should be directed. If a tape manager is installed and the SQLAFUTX EXEC has been coded, these messages will not be issued.
- To define the scan criteria in the variables **Database, Table_Creator, Table_Name, Command_Type, Auditing_Date, Auditing_Time, SQL_Userid, VM_Userid, Program_Creator, Program_Name, SQLcode** and **Terminal_id**.
- To define the eventual column names and values to be used in the scan in the variables **Column_name.1** to **Column_name.4** and **Column_value.1** to **Column_value.4**.
- If a variable need not be assigned, it should be set to blank.

Issue an XEDIT **file**.

6.4.2 Executing the scan parameter file in your own machine

- Issue the command **[EXEC] SQLAFBAT <yourname> SQLAFBAT A** to start the archive tape scan.
- If an execution time was specified, SQLAFBAT will pause until that time.
- Unless you are using a tape manager, messages will be issued to attach a tape drive and mount the archive tape(s).
- The output of the scan is stored on your A-disk in a file named **AFyymmdd SLhhmmss**.
- To inspect the scan results, issue a CMS FILELIST and type **SQLAF** on the corresponding filelist line.

6.4.3 Submitting the scan parameters to an archive scan server

- To have your scan parameter file processed by the scan server, forward the file to the server, using a CMS SENDFILE command.
- The server returns the scan results as a virtual reader file named **AFyymmdd SLhhmmss**. Issue a CMS RECEIVE.
- To inspect the scan results, issue a CMS FILELIST and type **SQLAF** on the filelist line for **AFyymmdd SLhhmmss**.

For details about setting up an archive scan server, see page 36.

6.5 Initiate an audit log archive

When you request an audit archive, a message is sent to the Audit Processor to initiate the audit archiving procedure.

6.6 Shutdown the Audit Processor

A message is sent to the Audit Processor to end the pending work and terminate. Please note that all SQL commands will terminate with SQLCODE -901 as long as the Audit Processor is inactive.

6.7 Suspend database auditing

The suspend function will ask the name of the database for which auditing has to be suspended and transmit a message to the Audit Initiator. Unlike shutting down the Audit Processor, **SQL commands operating on a protected table will continue normally**. Therefore, this function should be used with caution, as it makes auditing temporarily non-operational.

6.8 Resume database auditing

The resume function will ask the name of the database for which auditing has to be resumed and transmit a message to the Audit Initiator.

6.9 XEDIT the SQLAF RULES dataset

This function allows you to modify the auditing rules. For a description of the RULES statements, please refer to page 23.

You must restart the Audit Processor (the SQLAFP program) if the rules should take an immediate effect. If you don’t, the new rules will take effect at the next auto-restart of the Audit Processor (which occurs at the begin of each day).

6.10 Re-apply GRANTs in the SQLAF RULES

When GRANT commands have been added or modified in the SQLAF RULES dataset, it is not necessary to restart the Audit Processor to apply them (for all statements other than GRANT this is required).

The apply GRANT function will ask the name of the database involved and request the Audit Processor to re-process the GRANT commands in the SQLAF RULES.

6.11 List SQLAF RULES history

Whenever the Audit Processor recognizes that the SQLAF RULES file has been modified, it takes a copy of the modified rules on its A-disk, in a CMS file named **AFRULES *yyyymmdd*** where yyyymmdd is the date of modification.

When you request the RULES history, a filelist of all AFRULES files will be shown on your screen, so that you may XEDIT the individual history files.

6.12 Disable auditing for user

Selecting this function displays a panel to supply the name of the databaseserver and the DB2/VM userid for which auditing should be disabled. The command takes immediate effect, even if the specified user is actually running. Auditing remains disabled for that user until the user is enabled from the user interface or until the database server is restarted.

6.13 Enable auditing for user

Selecting this function displays a panel to supply the name of the databaseserver and the DB2/VM userid for which auditing should be enabled. The command takes immediate effect.

6.14 Inspect dataspace counters

When SQL/AF dataspace support has been enabled, this function will show, for the named database, the following dataspace counters:

- total number of auditing requests stored in the dataspace during the database session
- total number of stored auditing requests processed by the SQLAFP program
- number of times the spill file has been used
- number of times the dataspace full condition has been detected
- number of pages in the dataspace
- number of pages in the “control” queue
  (usually 1, unless a very large number of pages is in the request queue)
- number of pages in the “free” queue
- number of pages in the “request” queue[14]
- maximum number of pages ever allocated to the request queue
  (may be used to monitor the defined dataspace size)

[14]Even when all stored requests have been processed, the number of pages in the request queue will be greater than zero, because processed pages are reclaimed and inserted in the free queue at the next auditing request.

6.15 Viewing saved log scan files

The log scan function allows to save a current log selection set to a CMS file which is named **AFyymmdd SLhhmmss**, where yymmdd and hhmmss stand for the date and the time of saving.

To view a saved selection file, issue a CMS filelist command and type **SQLAF** on the line containing the filename. This will invoke the SQL/AF editor to display the saved log rows.

7 Utility Programs

7.1 SQLAFCMD

The SQLAFCMD program is used to execute the ARCHIVE, SHUTDOWN, SUSPEND and RESUME functions of the interactive user interface from a CMS EXEC. The syntax of the SQLAFCMD command is as follows:

SQLAFCMD
{ARCHIVE|SHUTDOWN|SUSPEND|RESUME | NEWRULES | S_AUDIT2 | S_AUDIT1}
[DATABASE database]

7.1.1 Archive the audit log

To archive the current audit log, enter the command:
SQLAFCMD ARCHIVE

7.1.2 Shutdown the Audit Processor

To terminate the Audit Processor, enter the command:
SQLAFCMD SHUTDOWN

7.1.3 Suspend auditing

To temporarily suspend auditing for a designated database, enter the command:
SQLAFCMD SUSPEND DATABASE database-name

Unlike shutting down the Audit Processor, **SQL commands operating on a protected table will continue normally without being audited**. Therefore, this function should be used with caution, as it makes the auditing function temporarily non-operational.

7.1.4 Resume auditing

To resume auditing for a designated database, enter the command:
SQLAFCMD RESUME DATABASE database-name

7.1.5 Re-apply SQLAF RULES

To re-apply the rules for a designated database, enter the command:

SQLAFCMD NEWRULES DATABASE database-name

This command causes a modified SQLAF RULES to be processed by the Audit Processor. If new tables were added to the rules, they will be audited as soon as the NEWRULES command has been processed.

7.1.6 Switch to the secondary log

The command **SQLAFCMD S_AUDIT2** will switch to the secondary audit log when all R/W LUWs currently in progress have been terminated. Audit information for new LUWs are written to the secondary log.

The S_AUDIT2 command waits until the Audit Processor has finished writing to the primary log.

S_AUDIT2 ensures that the primary audit log contains audit entries for closed LUWs only. The primary log can then be used as input for the archive command or for table synchronization purposes.

7.1.7 Switch to the primary log

The command **SQLAFCMD S_AUDIT1** erases the primary audit log, copies all records from the secondary log to the primary log and erases the secondary log. New audit information is written to the primary log.

An implicit S_AUDIT1 occurs at completion of the archive function.

Notes

- You should not prefix the SQLAFCMD command with the word EXEC, since SQLAFCMD is a MODULE.
- The suspend and resume requests are directed to the Audit Processor. This means that the Audit Initiator still uses the auditing dataspace or IUCV communications to forward requests to the Processor, which ignores them. It is usually better to suspend the Audit Initiator, by emitting an SQLMFCMD AUDITOR DISABLE/ENABLE. (as is done by the interactive suspend function).

7.2 SQLMFCMD

The **SQLAFCMD** messages described in the previous section are forwarded to the Audit **Processor**. You may also send following control requests to the Audit **Initiator** component executing in the database server machine:

7.2.1 Enable / disable all auditing

EXEC SQLMFCMD dbn AUDITOR {ENABLE | DISABLE}

Enables or disables the Audit Initiator running in the database whose VMid is specified in **dbn**. The disable/enable request is a dynamic alternative for the static disabling provided through the IGNORE statement in SQLAF RULES, described on page 23.

7.2.2 Enable / disable auditing for a named user

EXEC SQLMFCMD dbn AUDITOR {ENABLE | DISABLE} {SQLID | VMID {n | *}}

- Enables or disables a named VM or DB2 userid in the database whose VMid is specified in **dbn**.
- Auditing can be disabled for multiple users concurrently.
- Specifying **SQLID *** or **VMID *** will enable or disable the executing VM userid.
- VMID has a broader scope than SQLID, since it disables all auditing for the virtual machine, whatever the SQLID that has been connected. SQLID disables auditing only when the named DB2 userid has been connected.
- SQLID and VMID cannot be specified on the same command.

Example

```
EXEC SQLMFCMD DBTEST AUDITOR DISABLE VMID *
EXEC SQLREORG DBSPACE nnn
EXEC SQLMFCMD DBTEST AUDITOR ENABLE VMID *
```

The above commands disable auditing during a DBspace reorganization.

Note

Since the above AUDITOR commands are executed in the DB2 server, you can also enter them on the server's console. For example, to disable auditing from the console, simply enter: **AUDITOR DISABLE**.

7.3 SQLAFRDQ

The SQLAFRDQ utility program removes one or all queued auditing requests from the auditing dataspace of a given database. The program should be used only when a non-recoverable SQL/AF software error causes a continuous restart of the Audit Processor, when it attempts to reprocess the same auditing request that raises the software error. (The databasename of the failing audit request is displayed on the console during SQL/AF abend.)

Before dequeuing the audit request(s), SQLAFRDQ prints the contents of the dequeued requests on the console spool file.

Request dequeueing should be performed as follows:

- logon the Audit Processor virtual machine
- bypass the PROFILE EXEC
- ensure that the SQL/AF and SQL/MF product disks are accessed
- on the CMS prompt enter the command **SQLAFRDQ <databasename> [ALL]**. To dequeue the first audit request only, omit the **ALL** argument (which is the recommended approach).
- execute the PROFILE EXEC to restart the Audit Processor

7.4 SQLAFXTR

The SQLAFXTR utility program extracts the UPDATE, INSERT and DELETE statements from the SQL/AF Audit Log and stores them, in executable form, into a named CMS file. This file can be submitted for processing to other programs, for example IBM's SQLDBSU.

The SQLAFXTR execution parameters are specified in a control file, whose name is specified during execution using the SQLAFXTR EXEC keyword.

SQLAFXTR is invoked as follows:

SQLAFXTR EXEC filename filetype filemode

Filetype defaults to **SQLAFXTR**
Filemode defaults to **A**

The SQLAFXTR control file may contain following control statements:

7.4.1 Required Control Statements

CONNECT [username IDENTIFIED BY password] TO SOURCE database

The source databasename is used to select the corresponding Audit Log entries for extraction. The named database should be active when extract is requested. If username is not specified, your default userid (that is your VM-id) is used.

OUTFILE filename [filetype] [filemode] [TERMINATOR x | NONE]

The OUTFILE statement designates the CMS file that will receive the extracted SQL statements. Filetype defaults to **SQLAFXTR**, filemode defaults to **A** and Terminator defaults to **;**

The optional argument TERMINATOR specifies the character to be appended at the end of the extracted SQL statements. With TERMINATOR NONE, no termination character is stored.

EXTRACT creator.tablename

Designates the table for which the SQL statements must be extracted. Both creator and tablename may be generic by using a trailing % sign. The specification %.% for instance will select all tables.

7.4.2 Optional Control Statements

FROM_LUW n

Requests SQLAFXTR to start extraction at the specified DB2 LUW-id. (The SQL/AF LogScan program can be used to view LUW-ids).

TO_LUW n

Requests SQLAFXTR to stop extraction at the specified LUW-id.

FROM_DATE date-expression

Requests SQLAFXTR to perform extraction for a specified date. Any valid DB2 date expression can be used. If specifying an absolute date, the date value must be enclosed in quotes.

TO_DATE date-expression

Used in combination with FROM_DATE, SQLAFXTR will perform extraction for the specified date range.

FROM_TIME time-expression

Requests SQLAFXTR to perform extraction using the specified time. Any valid DB2 time expression can be used. If specifying an absolute time, the time value must be enclosed in quotes.

TO_TIME time-expression

Used in combination with FROM_TIME, SQLAFXTR will perform extraction for the specified time range.

7.4.3 Example

This SQLAFXTR run will store all changes to the SQLDBA.CUSTOMERS table during the previous day into the CMS file CUSTOMER UPDATES A. The changes are stored as SQL statements.

Code following statements in the CMS file CUSTOMER CONTROL A:

```
CONNECT TO SOURCE <DB-name>
OUTFILE CUSTOMER UPDATES
FROM_DATE CURRENT DATE - 1 DAY
TO_DATE CURRENT DATE - 1 DAY
EXTRACT SQLDBA.CUSTOMERS
```

Execute the control file by means of the following command:

```
SQLAFXTR EXEC CUSTOMER CONTROL A
```

8 Archiving Considerations

8.1 Audit archiving

Archiving the disk-resident audit log dataset **SQLAF AUDIT1** is done by appending the dataset to the current audit archive tape. Depending on the expected volume of audit data generated and the characteristics of the tape devices used, the installation may choose to keep an archive tape per day, per week, per month or per year. The archive processor detects the begin of a new archiving period and will request a new archive volume during tape mount. In the course of an archiving period, the mount message will display the tape label of the current archive.

The archive processor uses the archiving period, as defined in the SQLAF OPTIONS dataset, to build the archive tape label, as follows:

Archive_cycle	Tape label generated
DAY	SQLAF-yyyy-mm-dd
WEEK	SQLAF-yyyy-mm-Wn (where n is a value between 1 and 5)
MONTH	SQLAF-yyyy-mm
YEAR	SQLAF-yyyy

The archive processor uses FILEDEF and LABELDEF commands to pass the label to the operating system, or to a tape manager, if present. The FILEDEF VOLID operand is not used (the volume label is not checked) and the FILEDEF DISP is set to MOD when appending to the current archive volume.

8.2 Scheduling the archive procedure

Ideally, audit archiving should be scheduled explicitly by operations after shutdown of all audited databases, by means of the SQLAF user interface program or the SQLAFCMD command.

However, when the primary audit log has been filled up for the [archive_limit] number of records or when the audit log is full, the Audit Processor (SQLAFP) will automatically schedule an archive operation. While archiving is in progress, audit requests will be stored temporarily in a secondary audit log (SQLAF AUDIT2). When archiving completes normally, the Audit Processor reverts to the primary log dataset.

The Audit Processor should be active when the archive request is issued.

8.3 Emergency archive procedure

If you need to perform an archive and the Audit Processor cannot be started (because its A-disk is full for example), apply the following procedure from a machine that is able to establish a read/write link to the audit logs.

1 To start the archive procedure enter : **EXEC SQLAFARC <name-of-audit-processor-machine>**. The SQLAFARC exec will request to attach a tape device and will link to the primary log dataset.

2 When archiving is complete, you will get error message SQLAFA010, which you may ignore.

3 When archiving has completed normally, acquire write access to the minidisks (or SFS directories) that contain the primary and the secondary log datasets.

4 Issue the following CMS commands:

ERASE SQLAF AUDIT1 n1
COPY SQLAF AUDIT2 n2 = AUDIT1 n1
ERASE SQLAF AUDIT2 n2

where:

n1 stands for the filemode of the primary log and
n2 stands for the filemode of the secondary log

8.4 Messages sent by the archive procedure

The archive procedure is executed by the SQL/AF Archive server machine (named SQLAFARC by default), using the SQLAFARC EXEC. The procedure issues the following messages, which should be routed to operations, by enabling the VM SCIF facility for the archive processor machine.

SQLAFA002 SQL/AF archive has been started
SQLAFA008 Please attach a tape drive to *SQLAFARC_userid* as 181
SQLAFA004 Mount scratch tape on 181 for archive *archive-tape-label*
(If a new archive cycle)
SQLAFA003 Mount *archive-tape-label* archive tape on 181
(If within an archive cycle)

9 Customizing the Auditing Facility

9.1 User Exit Program

When the Audit Processor SQLAFP finds the user exit program **SQLAFUXP EXEC** on its file search chain, it will invoke that EXEC before a record is written to the audit log. The exit can access all fields of the audit record and take appropriate action. There is no restriction regarding the processing allowed in the user exit. However, since the user exit runs as an extension of the Audit Processor, care should be taken not to engage in long-running transactions or to produce long wait states of any kind. For performance reasons, SQLAFP initialization loads the SQLAFUXP EXEC in storage and subsequently invokes the storage resident EXEC. This implies that changes to the disk-resident SQLAFUXP EXEC will take effect at the next (explicit or implicit) restart of SQLAFP. Please note that the command or agent screened by the user exit is not necessarily executing. The audit data flow from the database server to the audit processor (and the user exit) is an asynchronous one. When the queued audit request is being processed, the originating SQL command may have completed already.

9.1.1 Coding the exit

On entry, the SQLAFUXP user exit receives the audit logrecord fields as invocation arguments on the ARG statement. A sample exit is provided with the product as SQLAFUXP SAMPLE. The sample defines the invocation arguments on an ARG statement. User code may be inserted following that initial statement.

On exit from SQLAFUXP, returncode 0 will write the audit log record. To bypass writing the log record, specify returncode 8.

Note:
The user exit is invoked by the Audit Processor SQLAFP. Audit requests that are flushed by the user exit (returncode 8) have still been stored in the audit dataspace.

Names and contents of entry variables:

Variable Name	Contents
LDATE	the current date in DB2 format yyyy-mm-dd
LTIME	the current time as hh:mm:ss
LSQLID	the DB2 username performing the audited command
LVMID	the VM username performing the command
LDBN	the current databasename
LTABCR	the creator of the table affected by the audited command
LTABN	the name of the table affected by the audited command
LPROGC	the creator of the package containing the audited command
LPROGN	the name of the package containing the audited command
LSECT	the package section number of the audited command
LAGENT	the DB2 agent number performing the audited command
LLUWID	the DB2 LUW number for the audited command
LROWS	the number of rows inserted, deleted or updated by the current command
LSQLCODE	the SQLCODE resulting from the audited command
LTERMID	the terminal_id that executed the audited command
LCMND	the type of the audited command (SELECT, INSERT, UPDATE, DELETE, COMMIT, ROLLBACK)
LTEXT	text of the audited command

9.2 User Message Processing

Sometimes, applications may wish to send auditing related messages to a central message service. SQL/AF provides a generalized and simple service method, by providing the **SQLAFUMP EXEC** on the server side and the **SQLAFXCP** module on the client side (for those clients that cannot directly issue SMSG commands).

9.2.1 Message Server

The message server runs in a disconnected machine and invokes the SQLAFUMP EXEC from its profile. This EXEC connects to the CP Message service and executes all messages received from SMSG.

SQLAFUMP uses CMS piping facilities only, so that an installation may easily customize the server EXEC.

9.2.2 Message Client

Clients send a message to SQLAFUMP by using the command:

SMSG <VM_id> <text_of_command_to_execute>

where:

VM_id is the name of the machine running SQLAFUMP
text_of_command is the CMS command to be executed by SQLAFUMP. If the command calls an EXEC, the “EXEC” prefix should be used. VM commands should be prefixed by “CP”. The length of the commandtext is limited to 130 characters.

Example SMSG MSGPROC EXEC MY_EXEC my_arg1 my_arg2 ...

VSE or CICS applications may issue the SMSG command by invoking the module SQLAFXCP, which in fact allows them to execute any CP command.

The only parameter to be passed during the SQLAFXCP call is the address of a 130-byte command area.

PL/1 Example

```
DECLARE     SQLAFXCP ENTRY OPTIONS(ASSEMBLER);
DECLARE     COMMAND CHAR(130) INIT ('SMSG MSGPROC EXEC
            my_exec arg1 arg2 ...');
CALL SQLAFXCP (COMMAND);
```

Note: The “ENTRY” definition is PL/1 specific and does not apply to other languages.

10 Operational Considerations

10.1 Software Prerequisites

- VM/ESA Version 1 Release 1.0 or later
- DB2/VM Release 3.3 or later
- SQL/Monitoring Facility ("SQL/MF") Version 5.1 or later[15]

10.2 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 SQLCS 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.

10.3 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.

10.4 Shutting down the Audit Processor

There is usually no reason to shutdown the audit processor. If you decide to do so, you **must** use the shutdown function of the User Interface (see page 37). If you abruptly stop the processor by a logoff or a CMS IPL, the audit dataspaces may become corrupted, as the processor does not get the chance to save them to their mapdisks.

[15]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.

10.5 Automatic Audit Processor restart

The Audit Processor SQLAFP **explicitly** restarts itself at 01:00 a.m. or at the time specified by the SQLAFP startup parameter RESTART. Restart is performed by issuing an IPL CMS command, which executes the PROFILE EXEC, which again starts SQLAFP. At restart time, SQLAFP performs audit initialization for all databases occurring in the SQLAF RULES dataset.

When an abend occurs in the Audit Processor, a dump is sent to the virtual printer, the audit request queues are saved and an **implicit** restart is performed.[16] However, the Processor must avoid continuous abends and restart when repetitively processing the same audit request. Therefore, no restart is performed when an abend occurs during initial queue recovery.

If a persistent abend condition exists and you cannot correct it yourself (because it is due to an SQL/AF software error for instance) and if SQL/AF dataspace support has been enabled, execute the “request dequeuing” utility **SQLAFRDQ** which is described on page 58.

10.6 Long-running update jobs

Long-running batch-like 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 SQLCS CONFIG dataset (the auditor is invoked by the monitor)
- by means of the **PERIOD** statement in the SQLAF RULES file (page 23)
- by means of the **IGNORE** statement in the SQLAF RULES file (page 23)
- by using the **SQLAFCMD SUSPEND** command (page 55).
- by using the **SQLMFCMD DISABLE VMID/SQLID** command (page 57).
- by using the **enable / disable** functions of the **SQLAF** user interface (page 37).

[16] Contrarily to explicit restart, audit dataspaces are not cleared at implicit restart.

10.7 SQLCODE -901

To prevent applications from executing non-audited commands, the Audit Initiator forces an SQLCODE -901[17] for all SQL commands executed while the Audit Processor server machine is inactive, unless the **UNRESTRICTED** option has been specified on the SQLCS.CONFIG.AUDITOR statement.

If SQL/AF dataspace support has been enabled and 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. If a spill file has been defined (as described on page 17), SQLCODE -901 is presented when both the dataspace and spill file are full.

10.8 Modifying the hardcopy destination

The hardcopy function of the Audit Log Scan utility uses the SQLAFPFD EXEC to direct the hardcopy to either the virtual printer (the default) or to another destination (e.g. a CMS file). If you wish to alter the default hardcopy routing, XEDIT the **SQLAFPFD EXEC** and modify the FILEDEF for the **SQLAFPRT** dataset to suit your needs.

10.9 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.

[17] **-901**: "The current SQL command has failed due to a system error. You may continue use of DB2/VM."

10.10 Performance considerations

- For optimal performance and if your hardware allows it, you should enable the SQL/AF dataspace support, described earlier in this manual. With dataspaces, the database servers do not use IUCV to forward audit requests, but store them in an auditing dataspace, where they are processed asynchronously by the Audit Processor.

- When the SQL/AF dataspace support is not used, the **QUICKDSP** and/or **SHARE** options should be set for the virtual machine running the SQLAFP Audit Processor program. Depending on your configuration and paging subsystem, you may also decide to use the SET RESERVED command. The purpose of these measures is to ensure that the Audit Processor is dispatched as soon as an auditing request is transmitted. This will avoid response-time degradation in the database servers emitting an audit request. With dataspace support, communications between the audited database servers and the Audit Processor are asynchronous and cannot produce response time delays in the database servers.

10.11 Processing overhead due to auditing

The DB2/VM database servers subject to auditing will send an IUCV audit message to the Audit Processor, or store an audit request in the dataspace:

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

If the IUCV protocol is used, the Audit Processor notes the audit data in storage and immediately replies to the message sent by the database server. The time between the IUCV SEND and REPLY will add to the command response time in the database server. Further processing of the audit request by the Audit Processor continues asynchronously with processing in the database server.

If dataspaces are used for request communication, the only overhead incurred by the database server is the time needed to store the data in the dataspace[18]. Saving of the mapped dataspaces is performed by the Audit Processor.

[18]Dataspace pagefault handling is synchronous. However, since the auditing dataspace will be a relatively small request transit queue in most cases, the dataspace pages in use will tend to remain storage resident.

10.12 Audit control files

SQL/AF maintains auditing control information in CMS files which have the audited databasename as a filename and one of following filetypes: TABAUDIT, COLAUDIT, TABLIST, COLLIST, VUELIST and VCDLIST. These files exist on the A-disk of the Audit Processor and some on A-disk of the audited database servers. They should **not be erased** by the user.

10.13 Audit operations logfile

The Audit Processor maintains an operations logfile on its A-disk as a CMS file named **SQLAF LOG**. The log contains an entry for following events:

- start of the Audit Processor
- shutdown of the Audit Processor
- auto-restart of the Audit Processor
- abend of the Audit Processor
- audit log archive initiation (current number of records on the primary audit log is shown)
- audit log archive completion (current number of records on the secondary audit log is shown)

The log will keep the last 1024 events. Older events are automatically removed.

11 Space Estimates

11.1 Space requirements for the audit log

How much is logged?

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 command.[19] 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 (4K CMS blocks)	Number of log records held
one CMS block	8
one 3380 cylinder	1200
one 3390 cylinder	1440
one Megabyte of disk space	2048

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

If you are running VM/ESA Version 2 Release 1 or later, you should enable SQL/AF data compression.[20] This will reduce the space required for the audit log with 50 percent or more.

[19]The text of static SQL commands is also kept.

[20]The compression facility is described on page 79.

11.2 Space requirements for the audit queue

When SQL/AF dataspace support is not used, queueing a static command requires 1K of main **storage**. Dynamic commands require an additional 8K. PUT commands and commands using the WHERE CURRENT OF CURSOR clause need an additional 8K.

With SQL/AF dataspace support the request queue is maintained in a dataspace containing varying length queue entries. The size of the queue also depends on the **polling interval** used by the Audit Processor. This interval (the DSPOLL argument on the SQLAFP command) determines how often the queue is emptied. It has a default value of 1 second. With larger polling intervals, the number of audit requests "in transit" will increase and a larger dataspace may be needed.

11.3 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.

12 Data Compression

If you are running VM/ESA Version 2 Release 1 or later, SQL/AF may be requested to use VM/ESA Data Compression facilities when writing to the audit log. Compression will reduce the space required for the audit log with 50 percent or more.

12.1 Compressing the Audit Log

The VM/ESA Data Compression facility requires that **compression dictionaries** exist, before compression can be performed. These dictionaries are created using the **SQLAFCFB** EXEC which takes a non-compressed audit logfile as its input. Therefore, you must run SQL/AF without compression for some time, long enough to produce representative scan data (5000 records for instance).

To create the compression dictionaries, proceed as follows:

- Ensure that all audited databases are inactive.
- IPL the Audit Processor machine (the machine that runs the SQLAFP program) and bypass its profile.
- Ensure that you have access to the SQL/AF product disk.
- On the CMS prompt, type the command **SQLAFCFB**.

This command builds the compression dictionaries **SQLAF ACDICT** and **SQLAF AEDICT** on the A-disk. The build procedure make take several minutes to complete. About 50 4K CMS blocks will be needed for the dictionaries. After the dictionaries have been built, the existing log is compressed using the dictionaries.

When the SQLAFP program is started next, compression will be performed, due to the presence of the dictionaries.

When the audit log is inspected, data expansion will occur automatically while scanning the log.

Notes

- It is possible to re-execute the SQLAFCFB procedure later on. In this case the current compressed log must be expanded first, by executing the command **CFDCOMP SQLAF AUDIT1**.

- The compression dictionaries should **never** be deleted, as this would make the compressed data non-expandable and hence inaccessible. Therefore, the compression dictionaries should be backuped up as well.

- The compression dictionaries are read-only files, as far as compression and expansion is concerned. The dictionaries are written to by the SQLAFCFB procedure only.

12.2 Archiving a compressed log

Audit log data are always archived to tape in **non-compressed format**. This means that a compressed audit log will be expanded on the A-disk of the Archive machine, before being written to tape.

Consequently, additional A-disk will be needed during archiving. Since that space is needed for a short period only, it is recommended that the Archive Processor's A-disk be placed in an SFS pool.

Alternatively, in the procedure that calls the SQLAFARC exec, you may define a temporary disk and access it as filemode **W**. If a W-disk exists, SQLAFARC will use it instead of the A-disk during the expansion process.

12.3 Expanding a compressed log

After compression, an audit log record contains hexadecimal pointers to dictionary entries and is no longer readable. If you wish to process the log outside SQL/AF, you can expand it as follows:

- Link to the A-disk of the Audit Processor.
- Copy the log SQLAF AUDIT1 to your A-disk.
- Enter the command **CFDCOMP SQLAF AUDIT1 A**
  After the command, SQLAF AUDIT1 A has been expanded.

12.4 Determining the current compression efficiency

If you wish to know the current compression rate of the audit log, proceed as follows:

- Link to the A-disk of the Audit Processor.
- Copy the log SQLAF AUDIT1 to your A-disk.
- Enter the command **CFDCOMP SQLAF AUDIT1 A**
- Enter the command **CFCOMP SQLAF AUDIT1 A**

At completion of the CFCOMP program, the compression rate achieved with the current dictionaries is shown. If the efficiency is too low, you may decide to re-execute the dictionary build procedure, described above.

13 Audit logrecord layout

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

13.1 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)     VM user 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[21]
INTEGER     DB2/VM 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","PREP
")
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.

[21] The agent number is in DB2/VM internal format and includes the system agents: the first user agent has agentnumber 4.

13.2 “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/VM 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).

14 Messages

14.1 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/VM 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 CMS file system RC xxx when writing audit lists to disk

A CMS error has occurred when writing the audit control files to disk. If the returncode equals 13, provide more space on the A-disk of the audit processor. For other returncodes, call for software support. See also: *CMS Returncodes* on page 100.

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.

SQLAFP015 Queue restart in progress

Informatory message indicating that the previous audit session did not terminate normally. Pending auditing requests are being processed now.

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 SQLCS 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.

SQLAFP018 CMS rc xxx NUCXloading SQLAFUXP

A CMS error occurred when loading the SQL/AF nucleus extension needed by the SQL/AF user exit program.

Call for software support.

SQLAFP050 CMS file system RC xxx when writing to primary log

A CMS error other than disk full occurred for the primary log dataset. Request an audit archive using the **SQLAF** program. If the error persists, call for software support. See also: *CMS Returncodes* on page 100.

SQLAFP050 CMS file system RC xxx when writing to auxiliary log

If the RC does not equal 13, call for support. If the RC equals 13, the secondary log dataset is full and the audit processor will abnormally terminate. See also: *CMS Returncodes* on page 100.

If no audit archive is in progress, execute the emergency archive procedure, described on page 64.

SQLAFP051 Primary log is full.

Informatory message indicating that there is no more disk space to write SQLAF AUDIT1. An audit archive is automatically initiated. Audit requests received during archiving are written to the secondary log.

SQLAFP052 Switching to alternate log dataset

Informatory message issued after message SQLAFP051 or SQLAFP054.

SQLAFP053 Auto-archive has been initiated

Informatory message issued after message SQLAFP051 or SQLAFP054.

SQLAFP054 Archive limit reached

The ARCHLIM invocation argument has been reached. An automatic audit archive is initiated.

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.

SQLAFP057 Issuing XAUTOLOG for the Archive Processor

The Audit Processor could not reach the Archive Processor to communicate an archive request. The Audit Processor attempts to XAUTOLOG the Archive Processor. If successful, the archive request will be retried.

SQLAFP058 XAUTOLOG for the Archive Processor failed with RC xxx

The XAUTOLOG command returns RC xxx. The return code corresponds with the CP message number issued after an unsuccessful XAUTOLOG. For example: RC 3 corresponds to message HCP003E.

Ensure that the XAUTOLOG statement has been supplied in the VM directory entry of the Archive Processor.

SQLAFP059 RC xx writing SQLAF LOG. Log written to virtual punch.

An error occurred when writing the SQLAF LOG file (RC 13 for example which indicates that the A-disk is full). All current log entries are written to the VM/ESA virtual punch.

To recover the SQLAF log, logon the Audit Processor, make space on the A-disk, transfer the PUN entry to the virtual reader and issue the CMS command RECEIVE = = A.

SQLAFP099 HNDIUCV RC IS xxx

Call for support.

SQLAFP100 Timeout during IUCV CONNECT to <server>

If <server> is the name of a DB2/VM database server, an IUCV CONNECT request failed during SQLAFP initialization. Ensure that the database machine is active. Otherwise, call for software support

If <server> is the name of the SQL/AF archive processor, ensure that the archive server is active.

SQLAFP101 IUCV RC xx ON SEND PATH TO<server>

Ensure that the designated server is active. For a list of the most common IUCV returncodes, refer to page 99.

SQLAFP102 Timeout during IUCV SEND to<server>

Ensure that the designated server is active.

SQLAFP103 HNDIUCV RC IS xxx

Call for software support.

SQLAFP104 CMSIUCV RC xxx CONNECTING TO <server>

If <server> is the name of a DB2/VM database server, an IUCV CONNECT request failed during SQLAFP initialization. Ensure that the database machine is active. Otherwise, call for software support

If <server> is the name of the SQL/AF archive processor, ensure that the archive server is active.

For a list of the most common IUCV returncodes, refer to page 99.

SQLAFP800 RC xxx in ARCHIVE_COPY erase_audit1.
SQLAFP801 RC xxx in ARCHIVE_COPY copy_audit2.

Ensure that the audit processor has write access to both the primary and secondary log minidisk or directory. Otherwise, call for software support.

SQLAFP802 IUCV error issuing the archive request.

The audit processor attempted to forward an archive request to the archive server, but the server could not be reached. Start the archive server and restart the audit processor.

SQLAFP996 Continous abends. First request on <dbn> audit dataspace is cleared.

More than 3 abends have occurred during the same auditor session. The software assumes that the errors are caused by the contents of the first request record on the auditing dataspace.

Therefore, the Audit Processor calls the SQLAFRDQ program to remove the first record from the dataspace. Before deleting it, the record contents are printed on the console spool.

SQLAFP997 Unrecoverable error has occurred. Auditing has been disabled.

Message issued when an error has occurred that prevents continuation of the audit processor. Another error message will have preceded.

SQLAFP998 Archive Processor shutdown has been requested.

An SQLAF *shutdown* request has been received. Auditing will terminate when all the audit requests queued have been processed.

SQLAFP999 Archive Processor shutdown has been completed.

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

14.2 Audit Initiator messages

SQLAFI001 SQLAFCPY EXEC RC is xxx

The audit initiator was unable to copy the control files from the A-disk or directory of the audit processor. Ensure that the initiator is able to link to that minidisk or directory and that there is enough space on the A-disk of the database server.

SQLAFI002 SQLAFLNK unable to access auditor A-disk

See message SQLAFI001 above.

SQLAFI003 Checking for dataspace support ...
SQLAFI004 60 second timeout has been set.

A check is made for the existence of dataspace support, which implies sending an IUCV message to the Audit Processor. A timeout will occur after 60 seconds, if the Audit Processor does not respond.

SQLAFI004 Dataspace support enabled

Informatory message stating that a dataspace will be used for communication with the Audit Processor.

SQLAFI005 No dataspace support enabled

Informatory message stating that the IUCV protocol will be used instead of a dataspace for communication with the Audit Processor.

If your hardware has dataspace capabilities, you should enable the SQL/AF dataspace support by defining a mapdisk, as described in the chapter *Setting up the Audit Facility*.

SQLAFI099 HNDIUCV RC IS xxx

Call for software support.

SQLAFI100 Timeout during IUCV CONNECT to <server>

An IUCV CONNECT request for the audit processor failed. Ensure that the audit processor is active. Otherwise, call for software support

SQLAFI101 IUCV RC xx on SEND path to <server>

An IUCV SEND request for the audit processor failed. Ensure that the audit processor is active. Otherwise, call for software support

For a list of the most common IUCV returncodes, refer to page 99.

SQLAFI102 Timeout during IUCV SEND to <server>

An IUCV SEND request for the audit processor failed. Ensure that the audit processor is active. Otherwise, call for software support

SQLAFI103 HNDIUCV RC IS xxx

Call for software support.

SQLAFI104 CMSIUCV RC xxx CONNECTING TO <server>

An IUCV CONNECT to the audit processor failed. Ensure that the audit processor is active. Otherwise, call for software support

For a list of the most common IUCV returncodes, refer to page 99.

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.

SQLAFI400 Auditor has been enabled

Message confirming the AUDITOR ENABLE command.

SQLAFI401 Auditor has been disabled

Message confirming the AUDITOR DISABLE command.

SQLAFI402 AUDITOR command in error

Syntax error in the SQLMFCMD AUDITOR command.

14.3 Messages from Dataspace Support

Following messages are issued only when dataspace support has been enabled in SQL/AF, through the inclusion of MAPDISK statements in the SQLAF OPTIONS file.

The returncodes included in all following messages, except messages 204 and 205, are fully documented in the IBM publication SC24-5520-02 ***CP Programming Services*** chapter 22 ***VM Data Spaces CP Macros***. A summary of these returncodes is included following the message.

SQLAFD200 RC n creating auditing dataspace

An error occurred when the Audit Processor creates a dataspace for a database server.

RC 8 *Creating the new address space would cause the maximum number of address spaces allowed for your virtual machine to be exceeded.*

Action Update the XCONFIG ADDRSPACE directory statement (MAXNUMBER clause) for the virtual machine running the Audit Processor. A dataspace is created for each database audited.

RC 12 *Creating the new address space would cause the total size of all address spaces owned by your virtual machine to exceed the maximum permitted.*

Action Update the XCONFIG ADDRSPACE directory statement (TOTSIZE clause) for the virtual machine running the Audit Processor. The size of the dataspaces created for the audited databases depends on the total number of 4K blocks in all MAPDISKs declared in the SQLAF OPTIONS file.

RC 16 *The specified address-space name contains invalid characters.*

Action Call for software support.

RC 20 *The specified size for the new address space is out of range.*

Action Call for software support.

SQLAFD201 RC n after ALSERV ADD

RC 4 *There are no unused entries in the host access list that can be used to establish the new ALE.*

Action Update the XCONFIG ACCESSLIST directory statement (ALSIZE clause) for the virtual machine running the Audit Processor or the database server.

SQL/AF dataspace support requires one additional ALE in each database server audited. The Audit Processor requires as many ALEs as there are audited databases.

RC 8 *The specified ASIT does not identify a currently-existing address space for which your virtual machine is authorized for the requested type of access.*

Action Call for software support.

SQLAFD202 RC n permitting auditing dataspace access

RC 4 *The specified ASIT does not identify a currently existing address space that your virtual machine owns.*

Action Call for software support.

RC 24 *The specified virtual machine was already authorized for access to the specified address space.*

Action This message should normally not occur but may be ignored.

RC 28 *The specified user ID or VCIT does not designate a currently logged-on or disconnected virtual machine.*

Action This message should normally not occur but may be ignored. The permit request will be issued again when the database is up.

RC 32 *Your virtual machine is not authorized to use the PERMIT function of the ADRSPACE macro.*

Action Call for software support.

SQLAFD203 RC n after ALSERV REMOVE

RC 4 *The ALET specified by the ALET operand does not designate an ALE in either the valid or revoked states.*

Action Call for software support.

RC 12 *The ALET specified by the ALET operand contains invalid bit settings.*

Action Call for software support.

SQLAFD204 Auditing dataspace is full

No more space found in the auditing dataspace (and in the auditing spill file, if one was defined). 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.

Use the **Audit dataspace counters** menu option of the SQLAF interface program, to determine how many times the audit dataspace full condition has occurred.

Unless UNRESTRICTED auditing was requested on the SQLCS 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 by increasing the size of an existing MAPDISK or by adding another one.

SQLAFD206 RC n after MAPMDISK IDENTIFY

Action Ensure that the SQL/AF dataspace support has been setup correctly. Otherwise call for support.

SQLAFD207 RC n after MAPMDISK DEFINE

Action Ensure that the SQL/AF dataspace support has been setup correctly. Otherwise call for support.

SQLAFD208 RC n after MAPMDISK SAVE

Action Ensure that the SQL/AF dataspace support has been setup correctly. Otherwise call for support.

SQLAFD209 n dataspace pages allocated to <database>

Informatory message issued during initialization of the Audit Processor. It shows the number of 4K pages allocated to the auditing dataspace of the named database. The size of this dataspace equals the number of 4K blocks on all mapping disks defined for this database.

SQLAFD210 RC n querying dataspace <dataspacename>

Action Ensure that the SQL/AF dataspace support has been setup correctly. Otherwise call for support.

SQLAFD211 Dataspace initialization complete

Informatory message issued during initialization of the Audit Processor. It indicates that all auditing dataspaces have been successfully created.

SQLAFD212 SPILL_FM filemode missing or invalid. Spill_file support disabled.

A SPILL_FM statement has been coded in the SQLAF OPTIONS dataset, but the database name or the spill filemode is missing or invalid.

The support for spilling has not been enabled.

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 corresponding MAPDISK 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 SQLAF interface. The IGNORE statement, described on page 25, can be used to prevent future auditing for that job.

14.4 Audit logscan messages

SQLAFS001 OPEN error on archive tape dataset

A CMS error occurred when opening the archive tape. A CMS error message will have preceded.

SQLAFS002 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.

14.5 Archive Processor messages

SQLAFA000 SQL/Archive Processor at xxxxxxxx

Informatory message issued during startup of the archive processor.

SQLAFA001 SQLAFARC unable to access auditor A-disk.

The archive processor is unable to access the primary audit log on the A-disk or directory of the audit processor. Ensure that the processor is able to link to that minidisk or directory.

SQLAFA002 SQL/AF archive has been started.

Informatory message issued when an archive procedure is being started.

SQLAFA003 Mount <arch_id> archive tape on 181

An archive procedure is being started. The archived data will be appended to an existing archive tape.

Mount the tape labeled **<arch_id>**.

(**arch_id** stands for the archive tape label and is generated by the archive processor.)

SQLAFA004 Mount scratch tape on 181 for archive <arch_id>

An archive procedure is being started. The archived data will be written to a new archive tape, because a new archive period has started.

Mount a scratch tape and mark it with the label **<arch_id>**.

(**arch_id** stands for the archive tape label and is generated by the archive processor.)

SQLAFA005 Creating / extending archive <arch_id>

Informatory message indicating that archiving is in progress.

SQLAFA006 Archive <arch_id> has been created / extended

Informatory message indicating that archiving has normally completed.

SQLAFA007 MOVEFILE returncode <CMS_rc> creating archive <arch_id>

The CMS MOVEFILE command is used by the archive processor to write the audit log to tape. Refer to the CMS MOVEFILE command description for an explanation of CMS_rc.

SQLAFA008 Please attach a tape drive to xxxxx as 181

Archiving is starting. Attach a tape drive to the archive processor with virtual address 181.

SQLAFA009 HNDIUCV RC is xxx

Call for software support.

SQLAFA010 Timeout during IUCV CONNECT to xxx
SQLAFA011 IUCV rc xxx on SEND path to xxx
SQLAFA012 Timeout during IUCV SEND to xxx
SQLAFA013 HNDIUCV rc is xxx
SQLAFA014 CMSIUCV rc xxx connecting to xxx

The above messages may be issued at completion of the archive procedure. The archive processor then sends a completion message to the audit processor. This message fails, because the audit processor is not active. For a list of the most common IUCV returncodes, refer to page 99.

Before restarting the audit processor, you must execute the “post-archive” function yourself, as follows:

1 establish write access to the minidisks or directories containing the primary and secondary log datasets
2 issue the following CMS commands:

ERASE SQLAF AUDIT1 n1
COPY SQLAF AUDIT2 n2 = AUDIT1 n1
ERASE SQLAF AUDIT2 n2

where **n1** stands for the filemode of the primary log and **n2** for the filemode of the secondary log.

SQLAFA015 Expanding SQLAF AUDIT1

Data compression has been rquested for the auditor logfile. Before writing the log to tape, the compressed log is expanded on the A-disk of the Archive machine.

SQLAFA099 Returncode xxx during archive

An error has occurred during archiving. Another SQLAFA message will have preceded.

14.6 IUCV Returncodes

For a complete description of the IUCV returncodes, refer to the IBM manual **CP Programming Services** (SC24-5520-02) in the chapter **IUCV Function Descriptions**. Use the last 2 digits of the IUCV code displayed to locate the IPRCODE in the manual.

The recoverable IUCV error codes are the following:

1011	The target communicator is not logged on.
1012	The target communicator is logged on but is not enabled for IUCV.
1013	The MAXCONN value for the source communicator should be increased in the VM/ESA directory.
1015	The source communicator is not authorized (in the VM directory) to connect to the target communicator

For all other returncodes, call for software support.

14.7 CMS File System Returncodes for Read

1	File not found, disk not accessed, or insufficient authority.
2	Invalid buffer address.
3	I/O operation to a minidisk failed.
4	First character of file mode is illegal.
5	Number of records to read is equal to zero.
7	AFT is not marked with a record format of F or of V. If the file was not previously opened, this indicates that the file has an invalid record format.
8	Successful operation, but the buffer was too small to hold all of the requested data.
11	Number of records to read is not exactly one for a file with variable-length records.
12	No records were read because end of file was reached or because the position parameter specified a record number greater than the number of records in the file.
13	Found an invalid displacement in the AFT for a file with variable-length records (this indicates a coding error: it should not occur).
20	Invalid character detected in file name.
21	Invalid character detected in file type.
25	Insufficient free virtual storage available for file system control blocks
26	Position is negative, the number of records to read is negative, or position plus the number of records to process exceeds the file system capacity.
29	Storage group space limit reached.
30	Some error, other than those in this list of codes, occurred while accessing an SFS file. No rollback occurred.
31	Rollback occurred while trying to access an SFS file. The work unit ID on which the rollback occurred is the default work unit ID at the time the file was opened by the first operation to the file.

40	One of the following errors occurred: A required CSL routine was dropped. A required CSL routine was not loaded. There was an error in a user exit routine. There was an error calling the user accounting exit routine
42	The variable length record read is invalid.
48	File is empty.
49	External object cannot be opened.
50	File is in DFSMS/VM migrated status and implicit RECALL is set to OFF.
51	Error occurred during DFSMS/VM file recall processing.
55	APPC/VM error.
70	SFS file sharing conflict or minidisk file is already open by DMSOPEN or DMSOPDBK with an output intent.
80	I/O error accessing OS dataset.
81	OS read password protected dataset.
82	OS dataset organization is not BSAM, QSAM, or BPAM.
83	OS dataset has more than 16 extents.
84	Attempt to read a file on an OS or DOS formatted minidisk.
99	A required system resource is unavailable for one of the following reasons: There is insufficient virtual storage for the file pool server. The file pool server is unavailable. File is in migrated status and DFSMS is not enabled.

IBM Source: CMS Application Development for assembler
SC24-5453-02

If you can't find the returncode appearing in the SQL/AF error message in the above list, please refer to the Chapter **CSL / FSF Return Codes** in the IBM manual **VM System Messages and Codes.**

14.8 CMS File System Returncodes for Write

1	Not authorized to write to file.
2	Invalid buffer address.
3	I/O operation to a minidisk failed.
4	First character of file mode is illegal or disk not accessed.
5	Second character of file mode is illegal.
6	The last record number to be written is too large (more than 65535) to fit in a halfword and an extended plist is not specified.
7	Position specifies a record number that is more than one greater than the current number of records in a file with variable-length records.
8	Size of output buffer is not greater than zero or an attempt was made to write a null record to a file with variable length records.
11	FSCB is not marked with a record format of F nor of V.
12	Disk or directory not accessed R/W.
13	Disk is full.
14	Size of output buffer is not evenly divisible by the number of records for a file with fixed-length records.
15	Attempt to alter the record length of a file with fixed-length records.
16	Record format specified not the same as file.
17	Size of output buffer is greater than 65535 for a file with variable-length records.
18	Number of records to write is not exactly one for a file with variable-length records.
20	Invalid character detected in file name.
21	Invalid character detected in file type.
25	Insufficient free storage available for file system control blocks
26	Position specifies a negative record number or number of records to write is negative or position plus the number of records exceeds the file system capacity ($2^{31} - 1$) or logical block number computed by system exceeds the file system capacity ($2^{31} - 1$).
29	The storage group space limit was reached.

30	Some error, other than those in this list of codes, occurred while accessing an SFS file. No rollback occurred.
31	Rollback occurred while trying to access an SFS file.
38	File explicitly opened with read intent.
39	A disk is accessed as a read only extension of another, and a given file exists on the extension disk but not on the parent disk.
40	One of the following errors occurred: A required CSL routine was dropped. A required CSL routine was not loaded. There was an error in a user exit routine. There was an error calling the user accounting exit routine
49	External object cannot be opened.
50	File is in DFSMS/VM migrated status and implicit RECALL is set to OFF.
51	Error occurred during DFSMS/VM file recall processing.
55	APPC/VM error.
70	One of the following sharing conflicts occurred: The file is locked. The file pool server detected a deadlock. The file is open for write through SFS OPEN. The file is open for write by another user. You attempted to write to a file that is currently implicitly open for READ, but the file has been changed since it was originally opened. The minidisk file is already open by DMSOPEN or DMSOPDBK when issuing an FSWRITE.
80	I/O error accessing OS dataset.
81	OS read password protected dataset.
82	OS dataset organization is not BSAM, QSAM, or BPAM.
83	OS dataset has more than 16 extents.
84	Attempt to write a file on an OS or DOS formatted minidisk.

99	A required system resource is unavailable for one of the following reasons: There is insufficient virtual storage for the file pool server. The file pool server is unavailable. File is in migrated status and DFSMS is not enabled.

IBM Source: CMS Application Development for assembler
SC24-5453-02

If you can't find the returncode appearing in the SQL/AF error message in the above list, please refer to the Chapter **CSL / FSF Return Codes** in the IBM manual **VM System Messages and Codes.**