IBM Informix Trusted Facility Guide

IBM Informix Extended Parallel Server, Version 8.4
IBM Informix Dynamic Server, Version 9.4

March 2003
Part No. CT1TBNA

Note:
Before using this information and the product it supports, read the information in the appendix entitled "Notices."

Table of Contents

Chapter 2 Audit Administration

Introduction

In This Introduction

This introduction provides an overview of the information in this manual and describes the conventions that this manual uses.

About This Manual

This manual documents the secure-auditing facility of the database server. It provides information on how to set up and administer audit trails, extract and interpret audit records, and use SQL utilities and statements for audit analysis. It also helps you avoid the misuse of administrative tools that could compromise security.

This manual is not a computer-security or trusted-facility-administration training manual. For detailed information on those topics, see the suggested material in "Related Reading" on page 14.

Types of Users

This manual is for the following users:

- Database server administrators
- Operating-system administrators
- Database administrators
- Users of Informix database servers who are interested in secure auditing

Before reading this manual, you should have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational database management systems (RDBMSs) or exposure to RDBMS concepts
- An understanding of system administration
- A familiarity with the SQL statements that pertain to the events that you want to audit

If you have limited experience with RDBMSs, SQL, or your operating system, refer to the *Getting Started Guide* for your database server for a list of supplementary titles.

Software Dependencies

This manual assumes that you are using IBM Informix Dynamic Server, Version 9.4, or IBM Informix Extended Parallel Server, Version 8.4.

Assumptions About Your Locale

IBM Informix products can support many languages, cultures, and code sets. All culture-specific information is brought together in a single environment, called a Global Language Support (GLS) locale.

This manual assumes that you use the U.S. 8859-1 English locale as the default locale. The default is **en_us.8859-1** (ISO 8859-1) on UNIX platforms or **en_us.1252** (Microsoft 1252) for Windows environments. This locale supports U.S. English format conventions for dates, times, and currency, and also supports the ISO 8859-1 or Microsoft 1252 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you need to specify the appropriate nondefault locale.

For instructions on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the *IBM Informix GLS User's Guide*.

Demonstration Databases

The DB-Access utility, which is provided with your IBM Informix database server products, includes one or more of the following demonstration databases:

- The **stores_demo** database illustrates a relational schema with information about a fictitious wholesale sporting-goods distributor. Many examples in IBM Informix manuals are based on the **stores_demo** database.

XPS

- The **sales_demo** database illustrates a dimensional schema for data warehousing applications. For conceptual information about dimensional data modeling, see the *IBM Informix Database Design and Implementation Guide*. ♦

IDS

- The **superstores_demo** database illustrates an object-relational schema. The **superstores_demo** database contains examples of extended data types, type and table inheritance, and user-defined routines. ♦

For information about how to create and populate the demonstration databases, see the *IBM Informix DB-Access User's Guide*. For descriptions of the databases and their contents, see the *IBM Informix Guide to SQL: Reference*.

The scripts that you use to install the demonstration databases reside in the **$INFORMIXDIR/bin** directory on UNIX platforms and in the **%INFORMIXDIR%\bin** directory in Windows environments.

New Features

Windows

The following changes affect Trusted Facility features on Windows systems:

- Changes to Windows message logging

  In previous releases, the database server message facility sent messages to the Windows Security Event log, and also to the database server log file (**%INFORMIXDIR%\%INFORMIXSERVER%.log**). Windows XP and later Windows versions do not support message logging by applications to the Security Event log. Messages are now written to the Windows Application Event log, and also to the database server log file.

- Changes to the Secure Auditing Facility

  In previous releases, secure auditing messages were to the Windows Security Event log. Auditing messages are now sent to a log file. The directory path can be specified by the **onaudit** utility, and has a default value of **%INFORMIXDIR%\aaodir**.

For a comprehensive list of new features for your database server, see the *Getting Started Guide*.

Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other volumes in the documentation set. The following conventions are discussed:

- Typographical conventions
- Icon conventions
- Command-line conventions
- Sample-code conventions

Typographical Conventions

This manual uses the following conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth.

Convention	Meaning
KEYWORD	All primary elements in a programming language statement (keywords) appear in uppercase letters in a serif font.
italics ***italics*** `italics`	Within text, new terms and emphasized words appear in italics. Within syntax and code examples, variable values that you are to specify appear in italics.
boldface ***boldface***	Names of program entities (such as classes, events, and tables), environment variables, file and pathnames, and interface elements (such as icons, menu items, and buttons) appear in boldface.
`monospace` *`monospace`*	Information that the product displays and information that you enter appear in a monospace typeface.
KEYSTROKE	Keys that you are to press appear in uppercase letters in a sans serif font.
♦	This symbol indicates the end of one or more product- or platform-specific paragraphs.

Tip: *When you are instructed to "enter" characters or to "execute" a command, immediately press* RETURN *after the entry. When you are instructed to "type" the text or to "press" other keys, no* RETURN *is required.*

Icon Conventions

Throughout this manual, several different types of icons identify text. This section describes these icons.

Comment Icons

Comment icons identify three types of information, as the following table describes. This information always appears in italics.

Icon	Label	Description
	***Warning:***	Identifies paragraphs that contain vital instructions, cautions, or critical information
	***Important:***	Identifies paragraphs that contain significant information about the feature or operation that is being described
	***Tip:***	Identifies paragraphs that offer additional details or shortcuts for the functionality that is being described

Feature, Product, and Platform Icons

Feature, product, and platform icons identify paragraphs that contain feature-specific, product-specific, or platform-specific information.

Icon	Description
UNIX	Identifies information that is specific to UNIX
Windows	Identifies information that is specific to Windows
IDS	Identifies information that is specific to IBM Informix Dynamic Server
XPS	Identifies information or syntax that is specific to IBM Informix Extended Parallel Server

These icons can apply to an entire section or to one or more paragraphs within a section. If an icon appears next to a section heading, the information that applies to the indicated feature, product, or platform ends at the next heading at the same or higher level. A ♦ symbol indicates the end of feature-, product-, or platform-specific information that appears within one or more paragraphs within a section.

Command-Line Conventions

This section defines and illustrates the format of commands that are available in IBM Informix products. These commands have their own conventions, which might include alternative forms of a command, required and optional parts of the command, and so forth.

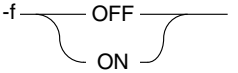
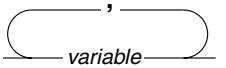
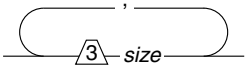
Dynamic Server supports a variety of command-line options. For example, the **onaudit** and **onshowaudit** utilities, which Chapter 4, "Utility Syntax," describes, require you to issue various commands, with one or more options that you can specify on the command line.

Each diagram displays the sequences of required and optional elements that are valid in a command. A diagram begins at the upper-left corner with a command. It ends at the upper-right corner with a vertical line. Between these points, you can trace any path that does not stop or back up. Each path describes a valid form of the command. You need to supply a value for each word that is in italics.

You might encounter one or more of the following elements on a command-line path.

Element	Description
command	This required element is usually the product name or other short word that invokes the product or calls the compiler or preprocessor script for a compiled IBM Informix product. It might appear alone or precede one or more options. You must spell a command exactly as shown and use lowercase letters.
variable	A word in italics represents a value that you must supply, such as a database, file, or program name. A table following the diagram explains the value.

(1 of 2)

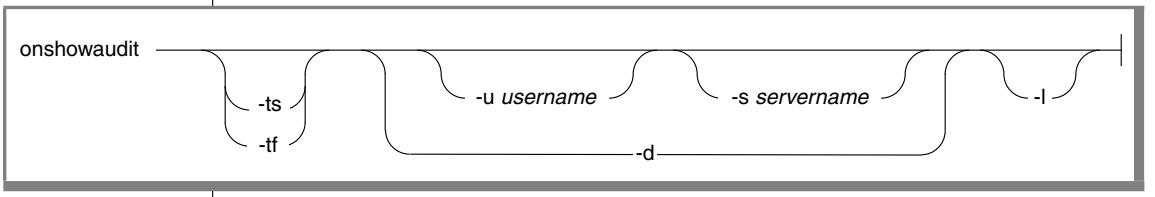
Element	Description
-flag	A flag is usually an abbreviation for a function, menu, or option name, or for a compiler or preprocessor argument. You must enter a flag exactly as shown, including the preceding hyphen.
.ext	A filename extension, such as **.sql** or **.cob**, might follow a variable that represents a filename. Type this extension exactly as shown, immediately after the name of the file. The extension might be optional in certain products.
(. , ; + * - /)	Punctuation and mathematical notations are literal symbols that you must enter exactly as shown.
' '	Single quotes are literal symbols that you must enter as shown.
Privileges p. 5-17 Privileges	A reference in a box represents a subdiagram. Imagine that the subdiagram is spliced into the main diagram at this point. When a page number is not specified, the subdiagram appears on the same page.
ALL	A shaded option is the default action.
	Syntax within a pair of arrows indicates a subdiagram.
	The vertical line terminates the command.
	A branch below the main path indicates an optional path. (Any term on the main path is required, unless a branch can circumvent it.)
	A loop indicates a path that you can repeat. Punctuation along the top of the loop indicates the separator symbol for list items.
	A gate (3) on a path indicates that you can only use that path the indicated number of times, even if it is part of a larger loop. You can specify *size* no more than three times within this statement segment.

(2 of 2)

How to Read a Command-Line Diagram

Figure 1 shows a Windows command-line diagram that uses some of the elements that the previous table lists.

Figure 1
Example of a Command-Line Diagram



To construct a command correctly, start at the top left with the command and then follow the diagram to the right, including the elements that you want. The elements in the diagram are case sensitive.

Figure 1 diagrams the following steps:

1. Type the word `onshowaudit` (to extract data from an audit trail).
2. You can type either `-ts` (to show only success audit records) or `-tf` (to show only failure audit records).
3. You can type `-d` (to use default values for the user and database server), or you can supply either or both of the -u `username` and `-s servername` options. To supply the -u `username` option, type -u followed by a valid username. To supply the -u `servername` option, type -s followed by a valid database server name.
4. You can type `-l` (to reformat extracted information for the **dbload** utility or convert it to an external file that you can load into an Extended Parallel Server database).
5. When you reach the terminator, your command is complete, whether or not you typed any of the options. Press RETURN to execute the command.

Additional Documentation

IBM Informix Dynamic Server documentation is provided in a variety of formats:

- **Online manuals.** The documentation CD in your media pack allows you to print the product documentation. You can obtain the same online manuals at the IBM Informix Online Documentation site at http://www.ibm.com/software/data/informix/pubs/library/.
- **Online help.** This facility provides context-sensitive help, an error message reference, language syntax, and more.
- **Documentation notes and release notes.** Documentation notes, which contain additions and corrections to the manuals, and release notes are located in the directory where the product is installed.

  Please examine these files because they contain vital information about application and performance issues.

UNIX

On UNIX platforms, the following online files appear in the **$INFORMIXDIR/release/en_us/0333** directory.

Online Files	Purpose
ids_tfm_docnotes_9.40.html **xps_tfm_docnotes_8.40.html**	The documentation notes file for your version of this manual describes topics that are not covered in the manual or that were modified since publication.
ids_unix_release_notes_9.40.html **xps_release_notes_8.40.html** **ids_unix_release_notes_9.40.txt** **xps_release_notes_8.40.txt**	The release notes file describes feature differences from earlier versions of IBM Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.
ids_machine_notes_9.40.txt **xps_machine_notes_8.40.txt**	The machine notes file describes any special actions that you must take to configure and use IBM Informix products on your computer. Machine notes are named for the product described.

♦

Windows

The following items appear in the **Informix** folder. To display this folder, choose **Start→Programs→Informix→ Documentation Notes or Release Notes** from the task bar.

Program Group Item	Description
Documentation Notes	This item includes additions or corrections to manuals with information about features that might not be covered in the manuals or that have been modified since publication.
Release Notes	This item describes feature differences from earlier versions of IBM Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.

Machine notes do not apply to Windows platforms. ♦

- **Error message files**. IBM Informix software products provide ASCII files that contain error messages and their corrective actions.

  UNIX

  To read the error messages on UNIX, you can use the **finderr** command to display the error messages online. ♦

  Windows

  To read error messages and corrective actions on Windows, use the **Informix Error Messages** utility. To display this utility, choose **Start→Programs→Informix** from the task bar. ♦

Related Reading

For a list of publications that provide an introduction to database servers and operating-system platforms, refer to your *Getting Started Guide*.

Compliance with Industry Standards

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL CAE (common applications environment) standards.

IBM Welcomes Your Comments

To help us with future versions of our manuals, let us know about any corrections or clarifications that you would find useful. Include the following information:

- The name and version of your manual
- Any comments that you have about the manual
- Your name, address, and phone number

Send electronic mail to us at the following address:

docinf@us.ibm.com

This address is reserved for reporting errors and omissions in our documentation. For immediate help with a technical problem, contact Customer Services.

Chapter

1 Overview of Auditing

In This Chapter

This chapter provides an overview of auditing and of auditing terminology for Dynamic Server and for Extended Parallel Server. It describes audit events, explains in detail how audit masks are configured and used, and indicates how to perform audit analysis. It also introduces the various audit administration roles.

Secure-Auditing Facility

Auditing creates a record of selected activities that users perform. An audit administrator who analyzes the audit trail can use these records for the following purposes:

- To detect unusual or suspicious user actions and identify the specific users who performed those actions
- To detect unauthorized access attempts
- To assess potential security damage
- To provide evidence in investigations, if necessary
- To provide a passive deterrent against unwanted activities, as long as users know that their actions might be audited

Important: *Make sure that users know that every action they perform against the database can be audited and that they can be held responsible for those actions.*

You cannot use auditing to track transactions to reconstruct a database. The database server has archive and backup facilities for that purpose. The *IBM Informix Backup and Restore Guide* explains these facilities.

Audit Events

Any database server activity that could potentially alter or reveal data or the auditing configuration is considered an *event*. The database server secure-auditing facility lets you audit and keep a record of events either when they succeed or fail, or simply when the activity is attempted. You can identify each audit event by a four- or five-letter event code called an *audit-event mnemonic*. Appendix A, "Audit Events," lists the audit-event mnemonics and describes the events that you can audit with the secure-auditing facility.

You can specify events that you want to audit in an *audit mask*. Auditing is based on the notion of audit events and audit masks.

Audit Masks

Audit masks specify those events that the database server should audit. You can include any event in a mask. The masks are associated with user IDs, so that specified actions that a user ID takes are recorded. Global masks **_default**, **_require**, and **_exclude** are specified for all users in the system.

Before you use auditing, you need to specify which audit events to audit. To specify audited events, add the events to the masks. You also need to perform other tasks, which Chapter 2, "Audit Administration," describes.

The database server does not provide auditing for objects or processes. For example, you cannot ask the database server to audit all access attempts on a certain object. You can, however, filter audit records from the audit trail based on objects with the audit-analysis tools, which Chapter 3, "Audit Analysis," describes.

Figure 1-1 represents a set of audit masks. The actual masks and their features are explained in "Audit Masks and Audit Instructions" on page 1-9.

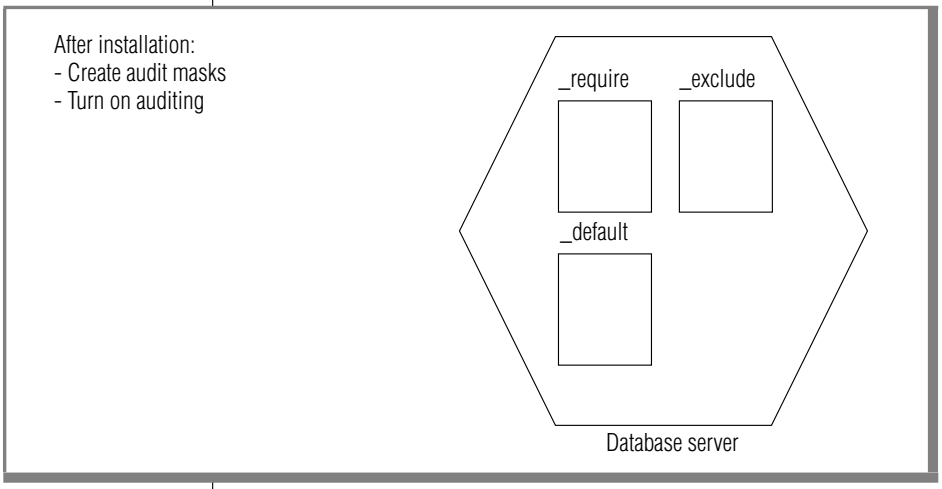


Figure 1-1
Audit Masks After Installation

After installation is complete, you can create the audit masks and turn on auditing.

Important: *If auditing is off, the database server does not audit any events, even if events are specified in the masks.*

In addition to the three masks that Figure 1-1 shows, you can specify *user masks* for individual users. User masks enable you to audit some users more than others and target different types of activities for different users. Except for the audit administrator who maintains the masks, a user cannot tell which events are being audited. For a description of user masks, see page 1-10.

You can also create *template masks* to create new user masks. For a description of template masks, see page 1-11.

Masks and their events are called *auditing instructions*, as Figure 1-2 shows. You have significant flexibility regarding the auditable facets of Dynamic Server. You can select anything from minimal audit instructions, in which no events are audited, to maximal audit instructions, in which all security-relevant database server events are audited for all users.

Defining masks:
- You must specify the events to audit within one or more audit masks.
- You can create masks for individual users.
- You can change the audit instructions during regular system operation.
- You can change a single mask during regular system operation.

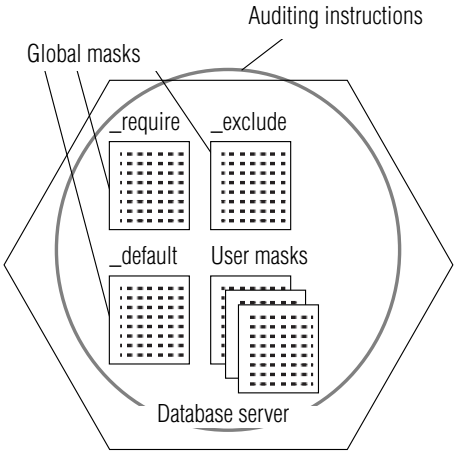


Figure 1-2
The Auditing Instructions

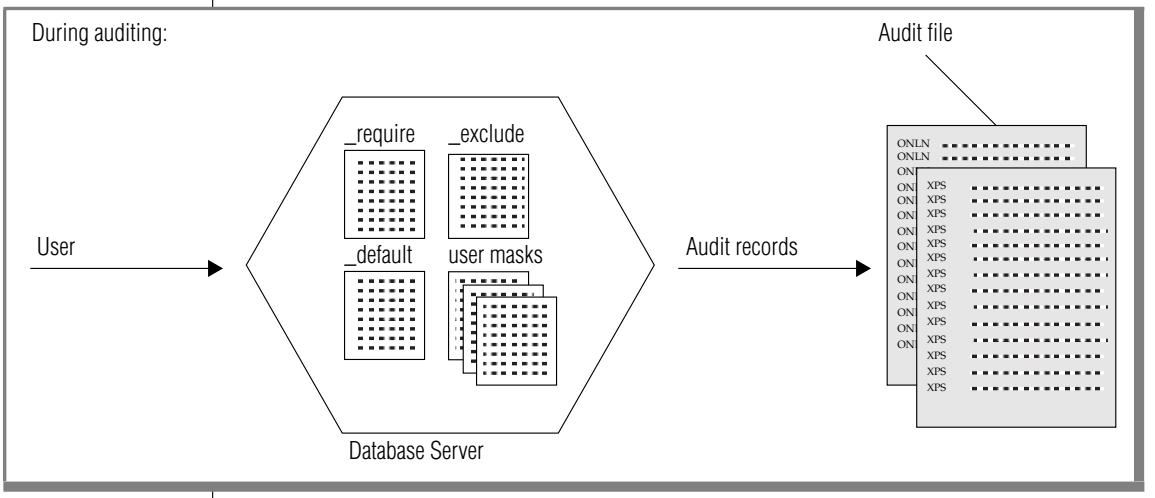
After you define the auditing instructions and turn on auditing, you can modify one or more audit masks as needs change and you identify potential security threats. For information on how to change audit masks, see Chapter 2.

Audit Process

When you turn on auditing, the database server generates *audit records* for every event that the auditing instructions specify, as Figure 1-3 shows. For UNIX, specify whether the operating system or the database server manages the audit records. For details, see "Types of Auditing" on page 1-18.

If you use database server-managed auditing, the database server stores the audit records in a file called an *audit file*, as Figure 1-3 shows. The collection of audit records makes up the *audit trail*. (The audit trail might consist of more than one audit file.) When operating-system-managed auditing is used on UNIX, the records are stored in an operating-system audit trail.

Figure 1-3
The Audit Process



An audit administrator needs to specify and maintain the *audit configuration*, which includes the following information:

- The audit mode
- How the database server behaves if it encounters an error when writing audit records to the audit trail
- For UNIX, the directory in which the audit trail is located
- For UNIX, the maximum size of an audit file before the database server or operating system automatically starts another audit file ♦

XPS

Audit files for Extended Parallel Server are stored locally on each coserver in the directory specified by the ADPATH parameter in the ADTCFG file or by the **onaudit -p** command. ♦

These topics are explained in "Audit Configuration" on page 1-17.

The database server generates audit records and writes them to the audit file or to an event log regardless of whether the client user that performs the audited action is local or remote. The database server includes both the user login and database server name in every audit record to help pinpoint a specific initiator and action.

IDS

In high-availability data replication (HDR), only the primary database server performs secure auditing and produces an audit trail. The **onaudit** utility runs on the secondary database server but does not audit any of the audit events. ♦

Audit Trail

Review the audit trail regularly. The database server offers a data-extraction utility, **onshowaudit**, that you can use to select audit data for specific users or database servers.

XPS

On Extended Parallel Server, you can select audit data for events on a specific coserver as well as for specific users. ♦

After you extract data, you can specify that it be formatted to load into a database for subsequent manipulation with SQL. "Audit Analysis" on page 1-26 explains this process.

Roles for Database Server and Audit Administration

The operating-system administrator (OSA) can set up the following roles for database server administration and audit administration, in addition to any administrative roles that your operating system might have:

- The database server administrator (DBSA) maintains and tunes the database server.
- An audit administrator can have either or both of the following roles:
  - Database system security officer (DBSSO), who specifies and maintains the audit masks
  - Audit analysis officer (AAO), who turns auditing on and off, sets up and maintains the audit configuration, and reads and analyzes audit-trail data

Although role separation provides more secure auditing, these roles are optional. Before the database server software is installed, the OSA, or whoever installs the database server, decides whether to have separate or combined DBSSO and AAO roles for audit administration and who should perform each role.

UNIX

On UNIX, to enable role separation the OSA must set the environment variable **INF_ROLE_SEP** to any positive integer before installing the database server software. If **INF_ROLE_SEP** exists, role separation is enabled. If role separation is not enabled, user **informix** can perform all administrative tasks, and no special groups are needed. ♦

Windows

IDS

On Windows, role separation is enabled through the **Role Separation** dialog box, which appears during installation. If the **Enable Role Separation** check box is checked in the **Role Separation** dialog box, the DBSA can specify different roles. ♦

For detailed information about roles and role separation, see "Role Separation" on page 2-7. For information about setting up role separation and creating a user group for each role, see your *Installation Guide*.

Audit Masks and Audit Instructions

As described in "Audit Masks" on page 1-4, an *audit mask* specifies a set of events to be audited when a user performs them. Audit events are derived from a combination of user and global masks. Appendix A lists the set of auditable events. The set of events is fixed, but you use masks to specify only the ones that you need to audit.

The following table lists four types of audit masks.

Mask Type	Mask Name
Individual user masks	*username*
Default mask	**_default**
Global masks	**_require** and **_exclude**
Template masks	*_maskname*

The following section describes the first three kinds of masks. For a description of template masks, see page 1-11.

User Masks

The global masks are always applied to user actions that are performed during a session in which auditing is turned on. Audit masks are applied in the following order:

1. An individual user mask or if none, the **_default** mask
2. The **_require** mask
3. The **_exclude** mask

When a user initiates access to a database, the database server checks whether an individual user mask exists with the same *username* as the account that the user uses. If an individual user mask exists, the database server reads the audit instructions in it first and ignores the **_default** mask. If no individual user mask exists, the database server reads and applies the audit instructions in the **_default** mask to that user.

In addition to default and individual masks, the database server reads and applies the audit instructions in the **_require** and **_exclude** masks. These masks are *global* because they apply to all users. Audit events in the **_require** mask are audited, even if they are not found in the **_default** or individual user masks. Audit events in the **_exclude** mask are not audited, even if the previously read masks specifically require them.

Important: *If the audit instructions of these masks conflict, the instructions in the last mask to be read are used. Masks are read in the following order: username, **_default**, **_require**, and **_exclude**.*

Users cannot tell if individual user masks exist for their accounts. Also, users do not need to do anything to enable auditing of their actions. Once an audit administrator turns on auditing, it operates automatically and users cannot disable it.

When the database server is installed, no audit masks exist. An audit administrator must specify all masks, including the default mask and the global masks.

Important: *Actions that the DBSA, an audit administrator, or user* ***informix*** *generally performs are potentially dangerous to the security of the database server. To reduce the risk of an unscrupulous user abusing the* ***informix*** *account, it is recommended that the actions of* ***informix*** *always be audited. This procedure is intended to prevent an unscrupulous user from using* ***informix*** *to tamper with auditing or from granting discretionary access to another unscrupulous user.*

Template Masks

As you become accustomed to the types of auditing that seem useful at your site, you might notice that certain auditing practices occur repeatedly. You can create *template* audit masks to help set up auditing for situations that recur or for various types of users.

For example, you might define a template mask called **_guest** and copy it to individual user masks for people who use your database server for a short time. You can copy a template mask to a user mask and modify it at the same time, perhaps turning off events that were audited in the template mask.

Important: *All template mask names must be unique, contain fewer than eight characters, and begin with an underscore (_). These naming rules distinguish template masks from individual user masks.*

You cannot create template masks with the following names because the database server already uses them:

- **_default**
- **_require**
- **_exclude**

When the database server is installed, no template masks exist. The number of template masks you can create is unlimited.

Audit Instructions

An audit administrator sets the audit instructions that the database server performs. The administrator must set an amount of auditing that is comprehensive enough to prove useful but not so exhaustive that it adversely affects system resources. When role separation exists, the DBSSO creates audit masks and the AAO configures mandatory auditing for the DBSA and the DBSSO. You can find advice on how to set the audit instructions in *A Guide to Understanding Audit in Trusted Systems* (published by the National Computer Security Center, NCSC-TG-001, June 1988).

This section suggests how to choose events to audit, how to set the audit instructions, and how the choices affect performance. For details of how to create and modify audit masks, see Chapter 2, "Audit Administration."

All the audit masks that the database server uses are stored in the system-monitoring interface (SMI) **sysaudit** table in the **sysmaster** database. The masks are updated automatically when the database server is upgraded to a newer version. Although information stored in the **sysmaster** database is available through SQL, you should use the **onaudit** utility for all audit-mask creation and maintenance. (See Chapter 4, "Utility Syntax.") Also, see the description of the **sysmaster** database in the *Administrator's Reference*.

Resource and Performance Implications

The amount of database server auditing enabled at any given time has a direct effect on operating-system resources and database server performance. Audit records that the database server generates are stored on disk. The greater the number of audit records generated, the more disk space required (for storage), and the greater the amount of CPU time required to process audit records (for storage, viewing, deletion, archiving, and restoration).

How system resources and performance are affected depends on these factors:

- Number of users/events audited
- Processor configuration
- System and user load
- Disk space
- Workload

For example, a system with parallel-processing capabilities, several gigabytes of available disk space, 64 users, and full auditing might experience little degradation in performance and a relatively small disk-space ratio for audit data. However, a single-processor configuration with 300 megabytes of available disk space, 10 users, and full auditing might experience significant system-resource degradation and relatively rapid disk-space consumption by the audit trail.

From a system performance standpoint, the greatest overhead is incurred when you audit all database server security-related events that all users perform. Full auditing could severely degrade system performance and response time as well as require a significant amount of disk space for audit-record storage (depending on the amount of database server user activity). However, full auditing provides the most audit information and thus reduces the security risk.

You can turn off auditing to eliminate the effect on system performance, but then auditing cannot contribute to system security. At a minimum, we advise that you audit the initiation of new user sessions.

The database server event that, if audited, has the most significant effect on system performance and disk space is Read Row (RDRW). In an established database that is primarily accessed by users who search for information, every row presented to every user generates an audit record. On a high-volume system, this quickly produces large numbers of audit records.

Suggested Minimum Set of Events to Audit

Although database server audit-record generation can adversely affect database server performance and resources, it is still advisable to perform more than minimal auditing. Audit enough events to detect security violations and attempts to circumvent security mechanisms. This section discusses some of the points to remember when you balance security needs with the performance and resource effects of different audit levels.

We recommend that you audit the following events for all standard database server users, at all times, with the **_require** audit mask:

- Open Database (OPDB)
- Grant/Revoke Database Access (GRDB), (RVDB)
- Grant/Revoke Table Access (GRTB), (RVTB)

IDS

For Dynamic Server, you should also audit the following events:

- Create Role (CRRL)
- Set Role (STRL)
- Set Session Authorization (STSA)
- Set Object Mode (STOM)
- Grant/Revoke Role (GRRL), (RVRL)
- Grant/Revoke Fragment Access (GRFR), (RVFR) ♦

The information contained in audit records that are generated when a user modifies discretionary access to an object is important. This information indicates what process changed the access, on what objects, and on whose behalf. In a typical environment, you can expect a low-to-moderate generation rate for audit records of this nature, which results in low disk-space consumption and minimal effect on database server performance.

It is also prudent to audit all database and table Open operations for all regular database server users. Auditing all Open operations indicates the general area within the database server where users are looking. Auditing these operations should not significantly affect database server performance because these operations are performed infrequently compared with other operations.

Creative attempts to circumvent the database server security policy are virtually impossible to detect if minimal or no auditing is performed for regular database server users. If you suspect a security violation or if the database server audit records reveal in the audit trail that a particular user exhibits unusual behavior, enable full auditing for that user. In this way, you can obtain a more complete picture of the activities of the user.

Special Auditing Considerations

Certain certification and accreditation organizations require that the installation process itself be audited. After configuring the operating system to accept audit data, the OSA should make sure that the AAO audits the actions taken during installation.

Level of Auditing Granularity

The Dynamic Server secure-auditing facility can audit the following events at the fragment level of granularity and shows additional information for fragmented objects:

- **Alter Table (ALTB).** The partition list that follows the alter-table operation is in the event record.
- **Create Index (CRIX).** The index can be fragmented; the event record includes fragmentation information.
- **Create Table (CRTB).** The table can be fragmented; the event record includes fragmentation information.
- **Delete Row (DLRW).** The partition and the record ID within the partition appear in the event record.
- **Insert Row (INRW).** The partition and the record ID within the partition appear in the event record.
- **Read Row (RDRW).** The partition and the record ID within the partition appear in the event record.
- **Update Current Row (UPRW).** The partition and the record ID within the partition appear in the event record.

Warning: *Use row-level auditing only when absolutely necessary. Row-level auditing slows the database server dramatically and fills audit directories quickly.*

For more information on the fields in an audit-event record, see Appendix A.

In addition, the database server audits the following events to the RESTRICT/CASCADE level:

- Drop Table (DRTB)
- Drop View (DRVW)
- Revoke Table Access (RVTB)

For more information on the corresponding SQL statements, see the *IBM Informix Guide to SQL: Syntax*.

Use of Various Masks

The **_require** mask can be a valuable tool because it audits every database server user for the events that are specified in this mask. You can use this mask to perform the bulk of the auditing. The **_require** mask enables you to make rapid changes to the auditing configurations for all users by adding or removing items from this one mask.

The **_exclude** mask is also useful. It is read last, so its contents take precedence over the instructions in the other masks. As the name implies, the audit events that you specify in the **_exclude** mask are excluded from auditing. This exclusion is true of every event, including those specified in the **_require** mask. The Read Row audit event, for example, is a good candidate for the **_exclude** mask. Read Row is a common event that can generate huge amounts of potentially useless data in the audit trail.

How you use the **_default** and individual user masks depends on the number of users and their activities. For example, if you have only a few users, you might want to give each one an individual mask. You might then use the **_default** mask to audit events that are initiated by users who do not normally use your database, and configure the **_default** mask with a high level of security. To offset any detrimental effects on system performance, set up less-comprehensive individual user masks for frequent users. Or, if you have many users and do not want to create many individual user masks, leave the **_default** mask empty and rely on the **_require** mask for most of your auditing.

Audit Configuration

The AAO can monitor the audit configuration, as Chapter 2 describes. Setting the audit configuration consists of performing the following tasks:

- Turning auditing on or off
- Specifying audit modes
- Using the **ADTCFG** file
- On UNIX, specifying database server-managed auditing or operating-system-managed auditing
- On UNIX, determining properties of the audit files ♦

UNIX

Sections that follow describe these topics.

Auditing On or Off

An audit administrator determines whether auditing is on or off. Auditing is turned off by default when the database server is installed. As Chapter 2, "Audit Administration," describes, the AAO can turn auditing on and off at any time, by using the **onaudit** utility, which Chapter 4, "Utility Syntax," describes. The database server can be in either online or quiescent mode for the changes to take effect.

When the AAO turns on auditing, all sessions are affected immediately. All user sessions that are started thereafter produce audit records.

Turning off auditing stops auditing for all existing sessions, and new sessions are not audited. If the AAO turns off auditing and then turns it on again while the database server is in online mode, existing sessions resume producing audit records.

Types of Auditing

When the AAO turns on auditing, the AAO can set the ADTMODE parameter in the **ADTCFG** file to specify the type and level of auditing.

Sections that follow briefly describe the types of auditing on UNIX and on Windows. For details, see "Changing the Audit Configuration" on page 4-12 and see Appendix B. For more information on auditing administration, see "Administrative Roles and Role Separation" on page 2-3.

UNIX

Auditing Modes on UNIX

If you act as the AAO, when you turn on auditing on UNIX, you can specify that either the database server or the operating system manage audit records. You set the ADTMODE configuration parameter to a number from 0 through 8 to specify the type and level of auditing.

For example, if you set the ADTMODE configuration parameter to 1 in your **ADTCFG** file on UNIX, database server-managed auditing turns on automatically when the database server initializes shared memory. After you turn on auditing, only the audit events defined in audit masks are recorded. (If you specify mandatory auditing for the DBSSO or the DBSA or both when you turn on auditing, audit records are generated for all events that are executed by the specified roles.)

The AAO sets the ADTMODE configuration parameter and specifies an error mode, in case an error occurs when an audit record is stored. The AAO must ensure that the operating-system audit facility is enabled if it is to manage the audit trail.

The OSA administers operating-system auditing and can configure auditing to monitor from single-user to system-wide events. Audit events are recorded in files in an audit trail. For the database server to use an audit trail that the operating system manages, the following criteria must be met:

- The operating system has an audit facility.
- The operating-system audit facility is enabled.
- The database server supports operating-system auditing for this platform.

If the operating-system audit facility manages audit records, the amount of database server auditing must also be acceptable to the operating-system administrator.

Windows

IDS

Auditing Modes on Windows

When you turn on auditing on Windows, you can set the ADTMODE parameter to `0`, `1`, `3`, `5`, or `7` in the **ADTCFG** file to specify the type and level of auditing.

For example, if you set the ADTMODE configuration parameter to `1` in your **ADTCFG** file, auditing is turned on automatically during database server initialization. After you turn on auditing, the database server records only the audit events defined in the audit masks.

The AAO configures auditing and specifies an error mode, in case an error occurs when an audit record is stored.

UNIX

Properties of Audit Files on UNIX

As "Audit Process" on page 1-6 describes, with database server-managed auditing on UNIX, the database server writes audit records to audit files in an audit trail. This section describes the audit files in more detail.

Location of Audit Files

The audit files are located in a directory that you specify with the **onaudit** utility or the ADTPATH configuration parameter in the **$INFORMIXDIR/aaodir/adtcfg** UNIX file.

XPS

Extended Parallel Server creates subdirectories for audit files in the path that you specify as the argument to the ADTPATH configuration parameter or as the argument to the **onaudit -p** command. The directory path that you specify must already exist on each node that hosts a coserver. For more information, refer to "Audit File Names" on page 1-20. ♦

If you change the audit path, the change takes effect immediately for all existing sessions. This feature enables you to change the directory when the database server is in online mode, which is useful if the file system that contains the existing audit files becomes full.

Keep the file system that holds the audit trail cleaned out so that ample storage space is always available.

New Audit Files

When the database server writes an audit record, the database server appends the record to the current audit file. If you bring the database server out of online mode and then put it back, the database server continues to use the same audit file. The database server starts a new audit file only under the following conditions:

- When the file reaches a specified size
- When you manually direct the database server to start a new audit file, as Chapter 2 describes
- When you start database server-managed auditing

The database server starts a new audit file at the default size of 10,240 bytes, which is the minimum size for audit files. (The **adtcfg.std** file might list a value of 50,000 bytes as a guideline.) You can change this file size at any time when auditing is on, even when the database server writes to an audit file, as Chapter 4 describes.

The optimal size for audit files depends on your configuration. Larger files contain more data, which results in fewer files to review. However, the trade-off is that large files are more difficult to manipulate.

Audit File Names

No matter how you start a new audit file, it follows the same naming convention.

In both Dynamic Server and Extended Parallel Server, the naming convention is *dbservername.integer*, where *dbservername* is the database server name as defined in the **ONCONFIG** file, and *integer* is the next integer. The series starts with `0`.

For example, if a new audit file is started for a database server **maple**, and the last audit file was saved in the file **maple.123**, then the next audit file is called **maple.124**. (If **maple.124** already exists, the next available number is used.) The names are unique to a specific audit directory, so you can have **auditdir1/maple.123** and **auditdir2/maple.123**, and so on.

XPS

Extended Parallel Server stores audit files locally on each coserver in a directory that you specify. For example, if you specify **/disk1/audit** as the location of audit files, the audit file directories and filenames would have the following form:

```
$disk1/audit/servername.coserver_id/servername.nnn
```

The variable *servername.coserver_id* combines the name of the database server defined in the **ONCONFIG** file and the number of the coserver that hosts the audit file. All audit files are stored locally on the coserver where the audited event occurs. Only one audit file directory exists for each coserver.

For example, if the database server is named **beech**, the audit files for coserver 3 are stored on a disk attached to the node that hosts coserver 3 in **/disk1/audit/beech.3** and the audit files in the **beech.3** directory have names such as **beech.111**, **beech.112**, and so on. If the node that hosts coserver 3 also hosts coserver 4, another directory named **beech.4** is created under **/disk1/audit** to contain audit files for events that occur on coserver 4. ♦

Windows

IDS

Windows Application Event Log

Windows systems provide an event-logging facility as a common repository for logging events and other useful information. The event-logging facility also provides a user interface to filter, view, and back up the information that is stored there.

In versions of Windows earlier than Windows XP, applications with appropriate permissions could write to the security log and to the system log. In Windows XP and later versions, however, applications cannot write to the Windows Security Event log. Auditing messages from the database server are now sent to a log file, whose directory path can be specified using the **onaudit** utility. The default pathname is **%INFORMIXDIR%\aaodir**.

Any messages that the database server writes to its log file are also written to the Windows Application Event log.

Windows

IDS

Windows Message Server

Dynamic Server for Windows runs as a service under the **informix** user account.

The Dynamic Server **Message Server** service communicates with the database server through the named pipes interprocess communications mechanism to receive information and to write it to the Windows Application Event log, as well as to the log file **%INFORMIXDIR%\%INFORMIXSERVER%.log**.

The database server starts **Message Server** when an instance of the database server first needs to write a message to the event log. **Message Server** does not terminate automatically when an instance of the database server terminates.

Error Modes for Writing to an Audit File

If the database server encounters an error when it writes to the audit file, it can behave in various ways called *error modes*. You can change the error mode, as "Setting the Error Mode" on page 2-11 describes, at any time during database server operation, even after an error occurs. See also the discussion of **onaudit** error modes in Chapter 4.

Halt Error Modes

When the database server is in a *halt* error mode (1 or 3), it does not allow the session that received the error to continue processing after it writes to the audit trail. The database server might even terminate the session or shut down, depending on the error mode. Descriptions of halt error modes follow:

- Mode 1: A thread is suspended but the session continues when the audit record is successfully written.
- Mode 3: The database server shuts down and the user session cannot continue.

Processing for the session does not continue until the error condition is resolved.

Continue Error Mode

When the database server is in *continue* error mode (0), it allows the session that received the error to continue processing after it writes to the audit trail. However, the audit record that was being written when the error occurred will be lost. The database server writes an error to the message log stating that an error made while writing an audit record has occurred.

If the error continues to occur, all subsequent attempts to write to the audit trail also generate messages in the message log, which can quickly grow very large.

Audit Configuration and the ADTCFG File

Configuration parameters in the **ADTCFG** file specify the properties of the audit configuration. For UNIX, and Windows, these configuration parameters are ADTERR, ADTMODE, ADTPATH, and ADTSIZE. For Extended Parallel Server, an additional UNIX parameter, ADTADMMODE, specifies whether to audit certain utility program events.

The pathname for the **ADTCFG** file follows.

Environment	ADTCFG Pathname
UNIX	**$INFORMIXDIR/aaodir/adtcfg**
Windows	**%INFORMIXDIR%\aaodir\adtcfg**

If you edit the **ADTCFG** file to change the audit parameters, the audit configuration is not changed until you reinitialize shared memory. If you use the **onaudit** utility to change the audit configuration, the changes occur immediately.

Changes made with **onaudit** are written to an **adtcfg.*servernum*** companion file. (SERVERNUM is a parameter in the **ONCONFIG** file, which the *Administrator's Reference* describes). An audit administrator must manually copy the changes from the **adtcfg.*servernum*** file to the **ADTCFG** file. The intent is to make it harder for the DBSA to start an instance of the database server with invalid audit parameters. For details on how to use the **onaudit** utility to configure the **ADTCFG** file, see Chapter 4.

Access to the Audit Trail

Standard users should not be able to view or alter audit files. The audit trail (that is, the audit files) should be accessed only with the **onshowaudit** utility, which has its own protection, as follows:

- With role separation on, only an AAO can run **onshowaudit**.
- With role separation off on UNIX, only user **informix**, a member of the **informix** group, or user **root** can run **onshowaudit**. ♦
- With role separation off on Windows, only user **informix** can run **onshowaudit.** ♦

UNIX

Windows

IDS

UNIX

Access to Audit Files on UNIX

The following characteristics control access to audit files in a UNIX environment and protect them from being accidentally read or destroyed:

Ownership: **informix**

Group ID: same as **$INFORMIXDIR/aaodir**

Permissions: **660**

Important: *The AAO should be careful when selecting the directory in which the audit files are stored (ADTPATH). The directories in the path must have adequate ownership and access permissions for the level of risk that the AAO allows. The default directory (**tmp**) probably does not have adequate protection.*

The following examples show the security configuration for UNIX audit files with no role separation:

aaodir

Ownership: **informix**

Group ID: **informix**

Permissions: **755**

aaodir/adtcfg.std

Ownership: **informix**

Group ID: **informix**

Permissions: **644**

The following examples show the UNIX security configuration with role separation:

aaodir

Ownership: **informix**

Group ID: ***<aao_group>***

Permissions: **755**

aaodir/adtcfg.std

Ownership: **informix**

Group ID: ***<aao_group>***

Permissions: **644**

Warning: *Because any account with the group ID of* ***informix*** *or superuser (****root****) ownership, or both, can access the audit trail, you must exercise care to protect these accounts and their passwords.*

Windows

IDS

Access to Audit Records on Windows

The following characteristics control access to the Windows audit file and protect it from accidental viewing or deletion:

Ownership: **informix**

Group ID: same as **%INFORMIXDIR%\aaodir**

The following examples show how to control access to the Windows audit file:

aaodir

Ownership: **informix**

Group ID: **Administrator**

aaodir\adtcfg.std

Ownership: database server administrator

Group ID: **Administrator**

Audit Analysis

The AAO performs audit analysis. This section explains the importance of audit analysis, how to prepare for it, some strategies for audit analysis, and how to react to a perceived security problem.

Importance of Audit Analysis

The database server audit mechanism is designed to both deter and reveal attempted, as well as successful, security violations. However, the audit data it generates is only as useful as the analysis and reviews performed on it. Never reviewing or analyzing the audit data is equivalent to disabling auditing altogether (and is, in fact, worse because auditing might reduce database server performance).

If, on the other hand, you routinely analyze and review the audit data, you might discover suspicious activity before a successful violation occurs. The first step to terminate any security violation is to detect the problem. If a database server violation should occur, the audit trail permits you to reconstruct the events that lead up to and include this violation.

Tip: *To play the greatest role in the security of your database server, watch the database server activity regularly.*

Become accustomed to the types of activity that occur at various times of day at your site. You become the expert on types of user activity when you perform the following actions:

- Review the database server security audit trail on a daily basis, or more frequently, if necessary.
- Note the types of activity that each user performs.

Periodically check the types of events that are audited versus the data that actually appears in the security audit trail to ensure that the audit facility is operating properly.

Your continual observance of the audit trail might be the only way to determine if some users browse through the database server. You might catch a user performing an unusual amount of activity at 2 A.M., a time of day when that user is not even at work. Once you identify a potential security anomaly, you can then investigate further to determine if anyone on the database server attempts to obtain unauthorized information, if a user misuses the database server, or if a user becomes lenient in self-regulated security enforcement.

Preparation for Audit Analysis

This section describes two methods to analyze database server audit records:

- The first method is simply to display audit data as it appears in the audit trail, which you can subject to your own audit-analysis tools. This method guarantees accuracy because no processing is done on the raw audit records.
- The second method converts the audit records into a form that can be uploaded into a table that the database server manages. You can then use SQL to generate reports based on this data. With the SQL-based method, you can create and use customized forms and reports to manipulate and selectively view audit data, which provides a flexible and powerful audit-analysis procedure. Be sure, however, that records are not deleted or modified from either the intermediate file or from the database prior to analysis.

Important: *The SQL-based procedure is more convenient but remains untrusted because users can use SQL data-manipulation statements to tamper with the records that are copied into a table.*

Both methods rely on a utility called **onshowaudit**, which Chapter 3 and Chapter 4 describe. For either method, you can extract audit events for specific users, database servers, or both.

Figure 1-4 shows the preparation process for both analysis methods. Chapter 3 explains each step in detail.

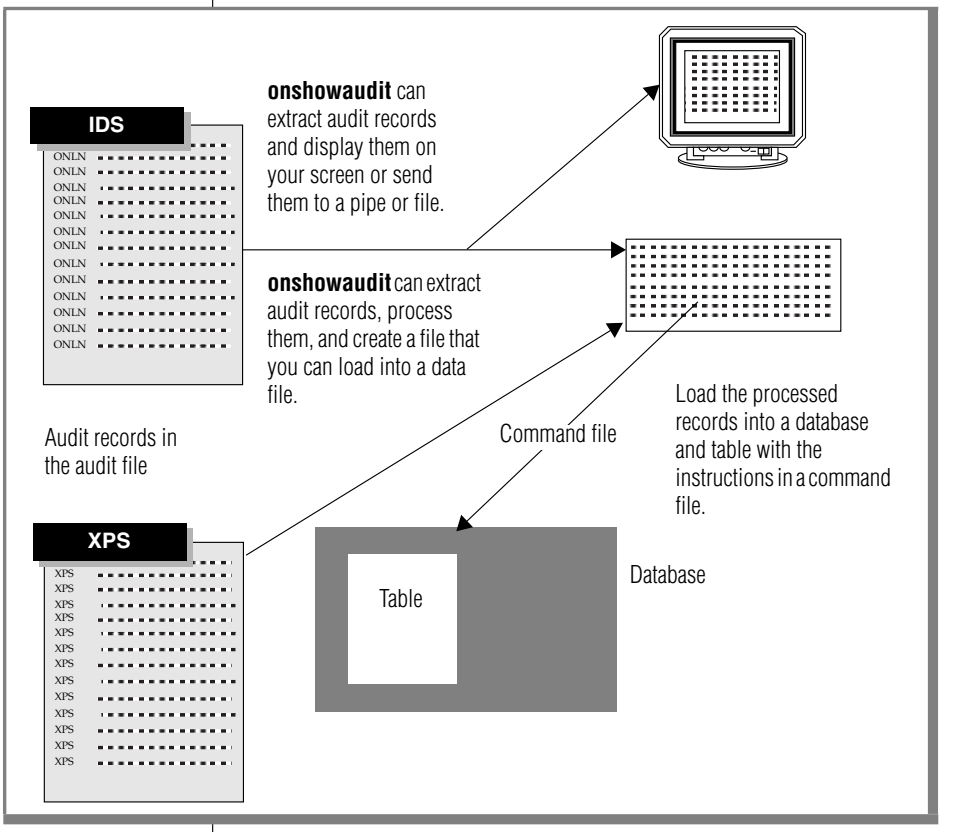


Figure 1-4
Preparing for Audit Analysis

To perform audit analysis, first have audit records in your database server or operating-system audit trail. The **onshowaudit** utility does not remove data from the audit trail. It only reads records from the audit trail and allows them to be viewed or manipulated with standard SQL utilities.

UNIX

When the following conditions are present on UNIX, records are in the operating-system audit trail:

- The operating system supports auditing.
- The database server supports operating-system auditing on this platform.
- For records in the operating-system audit trail, your database server must be registered as a protected subsystem with your operating system, as the UNIX machine notes file describes. (See "Additional Documentation" on page 12 of the Introduction.)
- Database server users have performed activities that generated audit records.
- Operating-system auditing is on. ♦

Windows

IDS

To clear or remove audit logs on Windows, delete the files that contain the audit trail. ♦

Strategies for Audit Analysis

The primary threat to database server security is unauthorized disclosure or modification of sensitive information. This section discusses those and other threats that might be discovered through audit analysis.

Event Failure

The audit records that indicate that an attempted database server operation failed are particularly important in audit analysis. The audit record could indicate, for example, that a user is attempting to give sensitive data to another user who does not have the correct UNIX permissions or Windows access privileges to access the data.

Event Success

Failed operations are the most common indicators of a security problem in the audit trail. Somewhat harder to find, but of equal security importance, is any successful but unusual activity for a particular user.

For example, a user who repeatedly creates and drops databases might be attempting to discover and exploit a covert channel to relay sensitive information to an unauthorized process or individual. Watch for a marked increase in the occurrence of database server events that would typically occur infrequently during normal database server use.

Perhaps a particular user who has never granted privileges suddenly shows a great deal of activity in this area, or perhaps a user who has never written large amounts of data into a database begins to generate hundreds of new records. You must determine the extent of the abnormalities (for example, the number of objects that this user accessed) and the possible severity of the compromise (for example, the importance of the accessed objects).

Insider Attack

An *insider attack* occurs when an authorized user with malicious intent obtains sensitive information and discloses it to unauthorized users. An unscrupulous user of this sort might not exhibit immediately recognizable signs of system misuse. Auditing is a countermeasure for this threat. Careful auditing might point out an attack in progress or provide evidence that a specific individual accessed the disclosed information.

Browsing

Users who search through stored data to locate or acquire information without a legitimate need are *browsing*. Browsers do not necessarily know of the existence or format of the information for which they are looking. Browsers usually execute a large number of similar queries, many of which might fail because of insufficient privileges. Auditing is a countermeasure for this threat. The behavior pattern makes browsers relatively easy to identify in the audit trail.

Aggregation

An *aggregate* is an accumulation of information that results from a collection of queries. An aggregate becomes a security threat when it comprises queries to objects that have little significance themselves but as a whole provide information that is considered more important than any component piece. The higher sensitivity of the aggregate results from the sensitivity of the associations among the individual pieces. Auditing is a countermeasure for this threat. As with browsing, careful auditing might point out an attack in progress or provide evidence that a specific individual accumulated the disclosed information.

Responses to Identified Security Problems

After you identify the user or users who are responsible for irregularities in the security audit trail, refer to your site security procedures. If your site has no security procedures regarding potential security breaches, you might consider the following actions:

- Enable additional auditing to further identify the problem.
- Shut down the database server to halt any unauthorized information flow.
- Develop a plan with the supervisor of the user to address the problem.
- Confront the specific individual.

In some cases, you might find that an otherwise authorized user is browsing a bit too widely on the database server. After some observation, you might want to talk with the supervisor of the user. It might not be wise to talk directly with an individual whose actions are being monitored.

You must ascertain whether a particular problem that is identified through the audit trail is actually someone attempting to breach security or just, for example, a programming error in a newly installed application.

The exact type of security irregularity that might occur and the specific action to take in response to it are not within the scope of this manual.

DBMS Security Threats

This section discusses responses to various kinds of security threats to the DBMS. For more information on various roles, see "Administrative Roles and Role Separation" on page 2-3.

Primary Threats

Primary threats to the security of a database server involve unauthorized disclosure or modification of sensitive information. To counter these measures, the DBSSO, DBSA, and OSA must ensure that all users of the DBMS are identified and authenticated before they are able to use or access the software or data.

To this end, all users must be associated with a known identity, such as one of the following identities:

- A valid login ID in the operating-system password file
- Membership in a valid group in the operating-system group file
- Membership in a valid group that can access the database

In addition, all users who attempt to access data must satisfy Discretionary Access Control (DAC) restrictions before access is granted. DAC uses SQL statements to specify which users can and cannot access data in the database. Access can be allowed or revoked at the following levels:

- Database level
- Table level
- SPL routine level
- Role level
- Fragmentation level

These countermeasures are adequate for legitimate use of the product when users attempt to access the data directly. They cannot, however, counter threats of confidentiality or modification to the data posed by illegitimate use of the product, such as if a privileged user abuses his or her permissions or access privileges.

Privileged Activity Threats

Improper or unchecked activity by users with privileged roles (DBSSO, AAO, DBSA, or OSA) can introduce security vulnerabilities and possible threats to the database server. Dynamic Server is carefully designed to give the DBSSO, AAO, and DBSA only the abilities needed to do their jobs. Nevertheless, these roles, as well as those of operating-system administrators, impart sufficient power that careless use of such power could result in breaches of security.

Database Server Administrator

The DBSA controls and monitors the database server and can configure role separation during database server installation. The countermeasure to a threat from the DBSA is independent scrutiny of the DBMS audit trail. The DBSSO can enable auditing of all DBSA actions, and the AAO can review DBSA actions in the audit trail.

Database System Security Officer

The DBSSO sets up DBMS audit masks for individual users. The countermeasure to a threat from the DBSSO is independent scrutiny of the DBMS audit trail because auditing DBSSO actions are enabled by the AAO.

Operating-System Administrator

A malicious OSA also poses a serious security threat because the OSA can violate the assumptions about the product environment and the methods that underpin its security functions. As with a DBSSO, the countermeasure to an OSA threat is independent scrutiny of the activities of the OSA, as recorded in the operating-system audit trail.

Audit Analysis Officer

The AAO reviews the DBMS audit trail. The countermeasure to this threat is to ensure that an AAO is authorized to view information that might be yielded when the database audit trail is reviewed. It is also important that the output of the **onshowaudit** utility be accessible only to an AAO and that manipulation of this output also be audited in the operating-system audit trail.

UNIX

Shared-Memory Connection Threats on UNIX

A shared-memory connection provides fast access to a database server if the client and the server are on the same computer, but it poses some security risks. False or nontrusted applications could destroy or view message buffers of their own or of other local users. Shared-memory communication is also vulnerable to programming errors if the client application explicitly addresses memory or over-indexes data arrays.

The OSA ensures that the shared-memory connection method is not specified in the configuration file for client/server connections. If the client and the server are on the same computer, a client can connect to a server with a stream-pipe connection or a network-loopback connection.

The pathname for the UNIX configuration file is **$INFORMIXDIR/etc/sqlhosts**.

For more information on shared-memory connections, see the *Administrator's Guide*.

Introduced Malicious Software Threats

A regular user might inadvertently execute malicious software, like a Trojan horse. This software, for example, might take one of the following actions:

- Attempt to copy data for subsequent access by an unauthorized user
- Grant DAC access privileges to an unauthorized user

Make all users aware of the dangers of executing software of unknown or untrusted origin. Further, take the following steps:

- All users should regularly check the DAC protection of the software with data that they own to ensure that access privileges have not been granted without their knowledge.
- Operating-system DAC should protect the software from modification by anyone other than authorized users.

Remote-Access Threats

When a user is granted DAC access privileges, the host computer of the user is not specified. Therefore, the user can gain access to the privileged data from any computer that is configured to connect to the host computer. As a result, a user might not be aware of having remote access to privileged data when the user grants another user direct access to that data. This situation could lead to data that is inappropriately accessed remotely.

Make sure that all users are aware that access privileges are granted to user names, with no dependencies on the origin of the remote connection.

Obsolete-User Threats

A user is identified by an operating-system user name or user ID or both. The DAC privileges and individual user audit masks of the software are based on the user name. At the operating-system level, a user account might be removed and this user name might become unassigned.

If any of the DAC privileges of the software or the individual user audit mask associated with that user name are not removed before the same user name is allocated to a new user, the new user inadvertently inherits the privileges and audit mask of the previous user.

To avoid this problem, have the OSA notify the DBSA when a user account is removed from the operating system. The DBSA can then perform the actions necessary to eliminate references to this name in the DBMS. These actions might involve revoking DAC privileges and removing an individual audit mask.

Untrusted Software Used in a Privileged Environment

Problems might occur if DBSAs execute untrusted software. This untrusted software could use the privileges of the DBSA to perform actions that bypass or disable the security features of the product or that grant inappropriate DAC access privileges.

The primary countermeasure to this vulnerability is to make sure that DBSAs do not execute software of unknown or untrusted origin. We further recommend that the operating-system access controls protect all software that DBSAs execute against unauthorized modification.

Distributed Database Configuration Threats

When you set up a distributed database, you configure two or more software installations. The configurations of these software installations could be incompatible.

A distributed database user might be able to gain access to data on a remote system with an incompatible configuration when that data would not be accessible to the same user directly on the remote system. In the worst case, the software could connect two systems that have an account with the same user name but are owned by a different user. Each user is granted the privileges of the other user at access of the database that resides on the host computer of the other user.

UNIX

When two UNIX workstations are connected, the OSA must ensure that accounts with user names in common are owned by the same user. ♦

Chapter

2

Audit Administration

In This Chapter

This chapter explains how to set up and administer auditing on your database server after the database server is installed and functioning properly. This chapter discusses the following topics:

- Administrative roles and role separation
- Setting up auditing
- Maintaining audit masks
- Maintaining the audit configuration, including turning off auditing

Administrative Roles and Role Separation

This section describes the main administrative roles involved in secure auditing:

- The database server administrator (DBSA)
- Audit administrator roles:
  - The database system security officer (DBSSO)
  - The audit analysis officer (AAO)

This section also touches on the roles and responsibilities of database administrators (DBAs), operating-system administrators (OSAs), system users, and privileged users. It tells how to set up role separation and provides guidelines on how to assign roles.

Database Server Administrator

The DBSA configures, maintains, and tunes the database server. The DBSA becomes involved with the security of a database server during installation. Your *Administrator's Guide* defines the overall role of the DBSA.

Someone who has the appropriate UNIX permissions or Windows access privileges to view all the data on a database server should perform this role. It is supported by a designated account and software designed to support DBSA tasks.

To use the administrative software designed for this role, the person who performs the role of the DBSA must log in to one or more designated accounts and meet access-control requirements.

Tip: *A DBSA is any user who belongs to the group* ***informix*** *(UNIX) or logs in as user* ***informix*** *(Windows), with or without role separation.*

Database System Security Officer

The DBSSO is a system administrator who performs all the routine tasks related to maintaining the security of a database server. These tasks include the following actions:

- Maintaining the audit masks
- Responding to security problems
- Educating users

The DBSSO performs these tasks with the **onaudit** utility. For information, see "The onaudit Utility" on page 4-4.

The DBSSO role is supported by a designated account and software. To use the audit tools, the users who fill the DBSSO role must log into the designated account and meet access-control requirements. After the DBSSO users meet the access-control requirements and use the administrative software, their actions can be audited.

Tip: *A DBSSO on UNIX is any user who belongs to the group that owns* ***$INFORMIXDIR/dbssodir****. On Windows, the administrator uses registry settings, through the* ***Role Separation*** *dialog box that appears during installation, to specify DBSSO users.*

Important: *The* ***onaudit*** *utility can create a potential threat to the security of the database server. An unscrupulous user can abuse a DBSSO account, for example, by turning off auditing for a specific user. To reduce this risk, all actions taken through* ***onaudit*** *should be audited.*

Audit Analysis Officer

The AAO configures auditing and reads and analyzes the audit trail. The AAO can specify whether and how auditing is enabled, how the system responds to error conditions, and who is responsible for managing the audit trail.

UNIX

For database server-managed auditing on UNIX, the AAO also determines the directory for the audit trail and the maximum size of each audit file. For operating-system-managed auditing on UNIX, the AAO should coordinate with the OSA how to read the data from the operating-system audit trail. ♦

The AAO can load the audit-trail data into a database server and use SQL to analyze it, either through a utility such as DB-Access or a customized application developed with an IBM Informix SQL API or application development tool.

The AAO performs these tasks with the **onaudit** and **onshowaudit** utilities, which Chapter 4 describes. If the AAO uses **onaudit** to change the audit configuration parameters during a database server session, the new values are written to the **adtcfg.***servernum* file for that instance of the database server.

The installation script for the database server creates a **$INFORMIXDIR/aaodir** UNIX directory or a **%INFORMIXDIR%\aaodir** Windows directory, which contains files that the AAO uses. These files include the **adtcfg** audit configuration file as well as the **adtcfg.std** file, both of which contain examples of valid definitions for audit configuration parameters.

The AAO needs appropriate UNIX permissions or Windows access privileges to view all the data in the database server to analyze events that might involve sensitive information. The AAO decides whether to audit all actions of the DBSSO and the DBSA.

Tip: *On UNIX, an AAO is any user who belongs to the group that owns* ***$INFORMIXDIR/aaodir****. On Windows, the administrator uses registry settings, through the* ***Role Separation*** *dialog box that appears during installation, to specify AAO users.*

Other Administrative Roles and Users

A number of other, more minor, roles might be involved in database server secure auditing. This section provides brief descriptions of these minor roles.

Database Administrator

A DBA manages access control for a specific database. A DBA cannot change database system modes, add or delete space, or maintain or tune the system. For information on the role and responsibilities of a DBA, see the *IBM Informix Guide to SQL: Tutorial*. For information on this and other database server roles and users, see your *Administrator's Guide*.

Operating-System Administrator

The OSA carries out responsibilities and tasks that the database server requires from the operating system. The OSA enables role separation, grants and revokes access to and from the database server if role separation is enforced, and adds new AAO, DBSSO, and DBSA accounts as necessary. In addition, the OSA coordinates with the DBSSO and AAO to perform various security-related functions of the database server, such as periodic reviews of the operating-system audit trail.

No special account exists for the operating-system needs of the database server, and no special database server protection mechanisms are associated with OSA tasks. For more information, refer to your operating-system documentation.

System Users

All operating-system accounts, including those for the DBSA, DBSSO, AAO, and the account called **informix**, potentially can use the database server. All users with accounts who want to use the database server must explicitly be granted access to the database server if role separation is configured to enforce access control on database server users. The DBSA can revoke that access at any time, whether or not role separation is enabled. For more information on granting or revoking access, see "Configuring and Enforcing Role Separation" on page 2-8.

Privileged Users

Privileged users are those users whom the database server recognizes as having additional privileges and responsibilities. These privileged users include the DBSA, DBSSO, AAO, and DBA. In addition, the users **informix** and **root** can also operate as any privileged user on database servers configured without role separation. Even with role separation, **root** can be a privileged user.

Role Separation

Role separation is a database server option that allows users to perform different administrative tasks. Role separation is based on the principle of separation of duties, which reduces security risks with a checks-and-balances mechanism in the system. For example, the person who determines what to audit (DBSSO) should be different from the person who monitors the audit trail (AAO), and both should be different from the person who is responsible for the operations of the database server (the DBSA).

Assigning Roles

This section provides general guidelines on how to assign people to accounts and give them access to perform roles. These guidelines should be amended to fit the resources and security policies of your site.

- Have one account for each person who performs a role.

  For example, if you have multiple users who perform the DBSA role, have each person work from a separate account. Establish a one-to-one mapping between accounts and users to make it easier to trace audit events to a single user.

- Have as few DBSA and DBSSO accounts as possible.

  The DBSA and DBSSO accounts can compromise the security of the database server. Limit the number of accounts that can disrupt the database server to lower the chance that an unscrupulous user can abuse a privileged account.

- Keep the DBSA and DBSSO roles separate.

  You might not have the resources or see the need to have different users perform the DBSA and DBSSO roles, nor does Dynamic Server strictly require this role separation. When you keep the DBSA and DBSSO roles separate, however, you constrain them to perform only those tasks that their duties specify and limit the risk of compromising security.

- Keep the AAO role separate from the DBSA and DBSSO roles.

  The AAO determines whether to audit all DBSA or DBSSO actions in the system. It is essential that someone with a role different from that of the DBSA or DBSSO be in charge of auditing configuration, so that all users, including the DBSA and DBSSO, are held accountable for their actions in the system. This constrains users to perform only those tasks that their duties specify and limits the risk of compromising security.

- Limit access to the account **informix** because it can bypass role-separation enforcement and other database server access-control mechanisms.

Configuring and Enforcing Role Separation

Role separation is configured during database server installation. The DBSA, or the person who installs the database server, enforces role separation and decides which users will be the DBSSO and AAO. To find the group for the DBSA, DBSSO, or AAO, look at the appropriate subdirectory of **$INFORMIXDIR** on UNIX or **%INFORMIXDIR%** on Windows.

UNIX

If the environment variable **INF_ROLE_SEP** is set, role separation is enforced and a group is specified for the DBSSO and the AAO as well as for standard users. If **INF_ROLE_SEP** is not set, user **informix** (the default) can perform all administrative tasks, and no special groups are needed.

You do not need to set **INF_ROLE_SEP** to a value to enable role separation. For example, in a C shell, issuing `setenv` INF_ROLE_SEP is sufficient.

For UNIX, role separation control is through the following group memberships:

- Users who can perform the DBSA role are group members of the group that owns the directory **$INFORMIXDIR/etc**.
- Users who can perform the DBSSO role are group members of the group that owns the **$INFORMIXDIR/dbssodir** directory.
- Users who can perform the AAO role are group members of the group that owns the **$INFORMIXDIR/aaodir** directory.

The **ls -lg** UNIX command produces the output that Figure 2-1 shows.

```
total 14
drwxrwx---  2 informix    <aao_group>  512 Nov 21 09:56 aaodir/
drwxr-xr-x  2 informix       informix 1536 Nov 30 18:35 bin/
drwxrwx---  2 informix  <dbsso_group>  512 Nov 30 10:54 dbssodir/
drwxr-xr-x 10 informix       informix  512 Nov 21 09:55 demo/
drwxrwxr-x  2 informix       informix 1024 Nov 30 11:37 etc/
-rwxrwxrwx  1 root           other    1234 Nov 21 09:56 filecheck*
.
.
.
```

Figure 2-1 *Example Output Showing Role Separation*

In Figure 2-1, the AAO belongs to the group **ix_aao**, the DBSSO belongs to the group **ix_dbsso**, and the DBSA belongs to the group **informix**.

Users must belong to the correct group to access the database server. To find the group for database users, you must look at the contents of the **$INFORMIXDIR/dbssodir/seccfg** file. For example, the contents of a typical **seccfg** file might be `IXUSERS=*`. This group setting means that all users are allowed to connect to the database server. If the file contains a specific name such as `IXUSERS=engineer`, then only members of the group **engineer** can gain access to the database server. ♦

Windows

IDS

For Windows, role separation control is through the **Role Separation** dialog box, which appears during installation, and through registry settings. If the **Enable Role Separation** check box is checked in the **Role Separation** dialog box, the DBSA can specify different roles. ♦

For more information on environment variables, see the *IBM Informix Guide to SQL: Reference*. For more information on configuring role separation, see your *Administrator's Guide*.

Auditing Setup

Auditing does not start automatically when the database server is first installed. Before any user actions are audited, the DBSSO or AAO must perform the following tasks to configure the database server for auditing:

- Specify events to audit in the default, user, and global audit masks (DBSSO)
- Specify how the database server should behave if an auditing error occurs when an audit record is written (AAO)
- Determine the desired level of auditing (AAO)
- Turn on auditing (AAO)
- Specify the directory where audit files are located (AAO)

Setting Up the Default and Global Masks

Before setting up default and global masks, the DBSSO needs to understand how the various masks work and what the implications are for different auditing instructions. Also, the DBSSO must understand which auditing events to place in which masks. For details, see Chapter 1.

Use the **onaudit** utility to add audit events to audit masks. Appendix A lists the audit events and their mnemonics. Chapter 4 shows the complete syntax for **onaudit**.

The following command shows how the Update Audit Mask and Delete Audit Mask audit events are added to the **_default** mask by their four-letter event codes, or mnemonics:

```
onaudit -m -u _default -e +UPAM,DRAM
```

You can add audit events to the **_require** and **_exclude** masks in the same way. For specifics, see Chapter 4.

All users who initiate a database session after this command is run (and auditing is turned on) are audited for the specified events.

UNIX

Specifying a Directory for the Audit Trail

As the AAO, when you turn on auditing, you specify that either the database server or your operating system manage audit records. If you choose to have your operating system control audit records, see your operating-system documentation for the location of those records.

If you specify that the database server store audit records, as Chapter 1 describes, the database server stores audit files in a file-system directory. You can specify the directory with the **onaudit** utility. For example, the following command specifies **/work/audit** as the UNIX file system in which the database server is to store audit files:

```
onaudit -p /work/audit
```

You can change the audit directory at any time.You can also set up the type of auditing and specify the directory with the **ADTCFG** file, which is described in Appendix B.

XPS

For Extended Parallel Server, the directory that you specify must exist on each node that hosts a coserver. For information about the naming conventions for the subdirectory and audit files, refer to "Location of Audit Files" on page 1-19. ♦

For more information about the **onaudit** utility, see Chapter 4.

Setting the Error Mode

As Chapter 1 describes, the database server has three actions that it can perform if an error occurs when writing to the audit trail: a *continue* error mode, and two levels of severity of *halt* error mode. Be sure that you, as the AAO, understand the implications of each error mode before you select one.

Use the **onaudit** utility or the **ADTCFG** file to set the error mode. For the **onaudit** syntax**,** see Chapter 4. For the ADTERR configuration parameter, see Appendix B.

The following **onaudit** command sets the error mode to *continue*. The database server processes the thread and notes the error in the message log.

```
onaudit -e 0
```

The following command sets the error mode to the most severe level of *halt*, in which the database server shuts down:

```
onaudit -e 3
```

Setting the Audit Level

The AAO or DBSSO configures the level of auditing in the system. The AAO monitors the audit trail and handles all audit-record management.

UNIX

If operating-system auditing is used on UNIX, before you can configure auditing, you must configure operating-system auditing to accept database server audit data. ♦

The DBSSO has significant leeway regarding the auditing level of the database server. For example, a minimal audit configuration might involve auditing only DBSSO actions, database server utilities, and the start of each new database server user session. A maximal audit configuration involves auditing all security-relevant database server events for all users.

The AAO and DBSSO should coordinate efforts to determine the auditing level. For instance, to audit the DBSA actions, the DBSSO would use masks for the DBSA accounts, and the AAO would set the audit mode with the **onaudit** utility or the **ADTCFG** file.

To ensure that the appropriate database server activities are monitored, review the audit records that are stored in the operating-system audit trail, database server audit files, or Windows event log. You must configure the database server to monitor these events.

You can reconfigure auditing as usage changes and potential security threats are identified. For the **onaudit** syntax, see Chapter 4. For information on the ADTMODE configuration parameter, see Appendix B.

Important: *Although database server audit-record generation might have a negative effect on database server performance and resources, you should perform more than the minimal database server audit. This additional audit improves the likelihood that you will detect security violations and any attempts to circumvent security mechanisms.*

If you perform minimal or no auditing for database server users, it is virtually impossible to detect creative attempts to circumvent the database server security policy. If someone suspects a security violation or a particular user exhibits unusual behavior, you should enable full auditing of the suspect user to get a complete picture of the user's activities.

Balance the security needs of your site and the performance and resource effect of different auditing levels. The auditing level at any given time has a direct effect on both the operating-system resources and the database server performance. The effect depends on the following factors:

- Number of users or events audited, or both
- Processor configuration
- System load (number of processes and users)
- Disk space
- Work load (types of processes performed)

Tip: *To specify disk space, use the Windows Event Viewer administration tool.*

For more information on database server performance considerations, refer to your *Performance Guide*.

Activating Auditing

Auditing is turned off by default when you install the database server. Use the **onaudit** utility to turn on auditing at runtime or set the ADTMODE configuration parameter in the **ADTCFG** file. If you use the **ADTCFG** file, the setting takes effect when the database server is initialized.

The following **onaudit** command turns on auditing:

```
onaudit -l 1
```

After you turn on auditing, auditing changes take effect immediately for all sessions.

The AAO can configure the database server to turn on auditing when shared memory is initialized when the ADTMODE configuration parameter is set to a number from `1` through `8` (UNIX) or to `1`, `3`, `5`, or `7` (Windows) in the **ADTCFG** file. For details on ADTMODE parameter values, see "Changing the Audit Configuration" on page 4-12 and Appendix B.

When the database server is initialized with auditing turned on, all user sessions generate audit records according to the individual, default, or global (**_require**, **_exclude**) mask in effect for each user.

To turn off auditing after it starts, see "Turning Off Auditing" on page 2-23.

Important: *It is recommended that the OSA always enable automatic auditing for the AAO in the operating system because the AAO can change the Informix DBMS audit configuration without being audited by the database server.*

Audit Mask Maintenance

You might want to change the auditing instructions as your auditing needs change. This section explains the following procedures, which you use to change audit masks:

- Creating audit masks
- Displaying audit masks
- Modifying audit masks
- Deleting audit masks

These tasks, which the DBSSO performs, apply whether the database server or your operating system administers the audit records.

Creating Audit Masks

You can create masks that more closely match the types of activities that individual users perform than do default and global masks. To create individual user masks, specify user IDs as mask names. To create template masks, preface the name of a mask with an underscore (_). Chapter 1 describes template masks and user masks.

You specify events in the mask when you create it, using the audit events from the alphabetical listing in the table "Audit-Event Mnemonics for IBM Informix Dynamic Server" on page A-2. You specify events for customized (template and user) audit masks the same way that you do for the **_default**, **_require**, and **_exclude** audit masks.

For example, you might want to create three template masks with different levels of security: **_low**, **_medium**, and **_high**. Alternatively, you might need just two templates for familiar and unfamiliar users that you copy to individual user masks: **_guest** and **_trusted**.

Creating a Template Mask

Use the **onaudit** utility to create template audit masks; Chapter 4 shows the syntax. The following example shows how to create a template mask called **_guest** with the audit events Create Database, Grant Database Access, and Grant Table Access:

```
onaudit -a -u _guest -e +CRDB,GRDB,GRTB
```

Creating a User Mask from a Template Mask

A mask that is used as the foundation for one or more other masks is referred to as a *base mask*. Once you create a template mask for a given user category, you can use it as the basis of masks for individual users, adding or removing only the audit events that differ for each user.

The following example creates a user mask for the user **terry**, based on the **_guest** template mask:

```
onaudit -a -u terry -r _guest -e -CRDB
```

The **terry** mask has the same audit events as the **_guest** mask, except for the CRDB (Create Database) audit event, which was removed.

Instead of template masks, you can also use existing user **_default**, **_require**, and **_exclude** masks as base masks.

Tip: *If you use a template or user mask as a base mask for another mask, the new mask inherits the events in the base mask.The new mask does not refer to the base mask dynamically. Future changes to the base mask are not reflected in other masks that might have been created or modified with that mask as a base.*

Creating a User Mask Without a Template Mask

You can create user masks without a template mask. The following example creates a mask for the user **pat** with the Show Table Statistics event and the failed attempts of the Alter Table event:

```
onaudit -a -u pat -e +SSTB,FALTB
```

For the syntax for creating a user mask and another example, see Chapter 4.

Adding One or More Masks Using an Input File

You can use the **onaudit** utility to add one or more masks to the mask table with instructions from a file that has the same format as the output of **onaudit -o**. The following command reads a file in **/work/audit_up** and adds audit masks to the mask table according to the instructions in that file:

```
onaudit -f /work/audit_up
```

Figure 2-2 shows an example of an input file. The syntax for the input file is explained in Chapter 4.

```
kickt    _secure1
jacks    -         +ADCK,SRDRW,GRDB,OPDB
pat      _secure2 +ALTB -CRTB,CRIX,STSN
jaym     -
johns    akee      -SALIX
```

Figure 2-2
Example Input File

The example input file in Figure 2-2 includes the following information:

- In the first line, the instructions specify auditing for user **kickt** in the new template **_secure1**.
- The second line creates a new mask called **jacks**, which contains the events Add Chunk (ADCK), successful attempts at Read Row (SRDRW), and all attempts at Grant Database Access (GRDB) and Open Database (OPDB).
- In the third line, the user **pat** is audited for all events that are specified in the template **_secure2**, and also for all attempts at Alter Table (ALTB), but not for attempts at Create Table (CRTB), Create Index (CRIX), and Start New Session (STSN).

- No template is specified for the target mask **jaym** in the fourth line, and no events are indicated; the mask is empty. (This prevents the **_default** mask from being applied to **jaym**.)
- In the fifth line, the target mask **johns** audits the same events as the mask **akee**, minus all successful attempts at Alter Index (SALIX).

Important: *Future changes to a base mask are not reflected in other masks that might have been created or modified with that mask as a base.*

An example of an audit mask input file, **adtmasks.std**, is provided in the **$INFORMIXDIR/aaodir** UNIX directory or in the **%INFORMIXDIR%\aaodir** Windows directory. The **adtmasks.std** file is intended only to serve as a guide to the DBSSO for how to set up an audit mask.

Audit masks do not work the same way as audit configuration parameters during initialization of the database server. (See "Audit Configuration and the ADTCFG File" on page 1-23.) Specifically, audit masks are not automatically read from a file and initialized.

Displaying Audit Masks

Use the **-o** option of the **onaudit** utility to display all the audit masks and the audit events that each mask contains. When you issue the **onaudit -o -y** command, the output (mask name, base mask, audit events) appears as follows:

```
_default      -      UPAM,DRAM
_require      -
_exclude      -
_guest        -      CRDB,GRDB,GRTB
terry         -      -CRDB
```

You can specify a mask as an argument to the **-o** option. The following example displays only the mask for user **terry**:

```
onaudit -o -u terry
```

A list of audit masks is helpful when you need to modify them. You can use the modified output as an input file to modify a single mask or groups of masks in a single batch. For more information, see "Modifying Audit Masks" on page 2-18. For the complete syntax of the **onaudit -o** option and a description of the output, see Chapter 4.

Tip: *If you use a base mask to create or modify a mask, the base mask itself does not appear in the* ***onaudit -o*** *output for the new mask. If a mask is created or modified with a base mask, it does not refer to the base mask.*

Modifying Audit Masks

The DBSSO can modify masks individually from the command line. (If you want to modify several masks at a time, you can create a new input file, change the appropriate masks, and reload them in the mask table.)

You can modify a single mask with the **-m** option of the **onaudit** utility. This option lets you use another mask as a base to add or remove individual audit events.

The following example shows how to modify the user mask **pat**. The **_guest** template mask forms a base from which a complete set of audit events is drawn. Settings for specific events from that file are then superseded by the events listed as arguments to the **-e** option.

```
onaudit -m -u pat -r _guest -e +ALTB,USTB
```

When you supply a base mask with the **-r** option, you replace all the audit events in the initial mask. When you change only a few events in a mask, you might not want to specify a base mask. For the syntax and another example of how to modify a mask, see Chapter 4.

Deleting Audit Masks

You can use the **-d** option of the **onaudit** utility to delete a single mask or all masks at once. The following example deletes the individual user mask for user **terry**:

```
onaudit -d -u terry
```

For the syntax of the **onaudit** utility, see Chapter 4.

Audit Configuration Maintenance

The AAO normally performs the following tasks to maintain the audit configuration:

- Displaying the audit configuration
- Changing the audit mode (including auditing specific roles)
- Changing the audit error mode
- Turning off auditing
- Starting a new audit file (including specifying a directory and maximum file size).

This section describes how to use **onaudit** to perform these tasks. For the syntax of the **onaudit** utility, see Chapter 4.

Displaying the Audit Configuration

You can display the current audit configuration with the **-c** option of the **onaudit** utility.

UNIX

Figure 2-3 shows output from the **onaudit -c** command on UNIX.

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright (c) IBM Corp., 1998 - 2003

Current audit system configuration:
   ADTMODE =  1
   ADTADMMODE=  0
   ADTERR =  0
   ADTPATH =  /tmp
   ADTSIZE =  20000
   Audit file = 64
```

Figure 2-3 *Example of Output from the* ***onaudit -c*** *Command on UNIX*

In Figure 2-3, the current audit system is configured as follows:

- ADTMODE is set to 1, which indicates that database server-managed auditing is on.
- ADTADMMODE is set to 0, which indicates that **onmode**, **oninit**, and **onstat** events are not audited.
- ADTERR is set to 0, which indicates a continue error mode.
- ADTPATH shows the default directory for audit files.
- ADTSIZE, which represents the maximum size of the audit file, is specified as 20,000 bytes.
- The number of the current audit file in the current audit directory is 64.

If you are user **informix**, you can also retrieve this information from the SMI **sysadtinfo** table in the **sysmaster** database. For details, see the *Administrator's Reference*. ♦

Windows

IDS

Figure 2-4 shows output from the **onaudit -c** command on Windows.

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright (c) IBM Corp., 1998 - 2003

Current audit system configuration:
   ADTMODE =   1
   ADTERR =    0
   ADTPATH =   %informixdir%/aaodir
   ADTESIZE =  50000
   Audit file = 0
```

Figure 2-4
Example Output from the onaudit -c Command on Windows

In Figure 2-4, the current audit system is configured as follows:

- ADTMODE is set to 1, which indicates that database server-managed auditing is on.
- ADTERR is set to 0, which indicates a continue error mode. ♦
- ADTPATH shows the default directory for audit files.

- ADTSIZE, which represents the maximum size of the audit file, is specified as 50,000 bytes.
- The number of the current audit file in the current audit directory is 0, meaning that no other audit file exists in the current series.

Starting a New Audit File

You can use a new file as the current audit file in the following ways:

- Use **onaudit -s** to change the maximum size of an audit file. If the audit file is already larger than the new size that you specify, the utility saves the current file and starts to write to a new one. The following example changes the default size to 20,000 bytes:

  ```
  onaudit -s 20000
  ```

- Use **onaudit -n** to start a new audit file without changing the maximum size. This option, which the following example shows, saves the current audit log to another file whenever you run it:

  ```
  onaudit -n
  ```

- Use **onaudit -p** to change the directory in which the database server writes audit files. The following example specifies **/work/audit** as the UNIX file system where the audit files are to be kept:

  ```
  onaudit -p /work/audit
  ```

  The directory that you specify must exist.

  XPS

  For Extended Parallel Server, the directory must exist on each node that hosts a coserver. ♦

Also, a new audit file starts every time that you start database-server-managed auditing.

You can use more than one flag at a time in an **onaudit** command. For the **onaudit** syntax to start a new audit file, change the audit-file size, or change the pathname of the audit directory, see "The onaudit Utility" on page 4-4.

UNIX

Changing the Audit Mode on UNIX

On UNIX, use the **onaudit** utility to change between operating-system-managed auditing and database server-managed auditing and to change the mandatory auditing of the DBSA or DBSSO or both. For example, to start basic operating-system-managed auditing, enter the following command:

```
onaudit -l 2
```

To start operating-system-managed auditing, which automatically audits the actions of the DBSA and DBSSO, enter the following command:

```
onaudit -l 8
```

Windows

IDS

Changing the Audit Mode on Windows

On Windows, use the **onaudit** utility to change levels of auditing by the database server and to change the mandatory auditing of the DBSA. For example, to start basic auditing, enter the following command:

```
onaudit -l 1
```

To start auditing and automatically audit the actions of the DBSA, enter the following command:

```
onaudit -l 5
```

Changing the Audit Error Mode

As Chapter 1 and "Setting the Error Mode" on page 2-11 explain, the database server behaves in one of three ways if it encounters an error when it writes to the current audit file. You can change the audit error mode with the **onaudit** utility. The following example directs the database server to suspend processing of the current thread and continue the write attempt until it succeeds:

```
onaudit -e 1
```

Turning Off Auditing

To turn off auditing, use the **onaudit** utility. The following example shows the command that turns off auditing:

```
onaudit -l 0
```

Warning: *Although auditing might be properly configured to audit the execution of a particular utility by a particular user, audit records might not be generated if the utility fails to execute for any of the following reasons:*

- *The user does not have the correct UNIX permissions or* Windows *access privileges to execute the utility.*
- *The user incorrectly specifies the command syntax of the utility.*
- *The utility cannot connect to shared memory.*

Chapter

3

Audit Analysis

In This Chapter

The importance of audit analysis cannot be stressed enough. This chapter explains the following topics:

- The format of audit records that the database server produces
- How to perform audit analysis with or without SQL
- How to extract audit information from the audit trail for quick viewing
- How to load that data into a database for analysis with SQL
- How best to perform audit analysis on the extracted audit information

This chapter applies whether you use the database server or your operating system to store and maintain the audit trail. An overview of the audit analysis process is in Chapter 1, "Overview of Auditing."

Audit-Record Format

The format for database server audit records has the following parts:

UNIX

- The first part is an operating-system audit header, if operating-system auditing is used on UNIX. The audit header contains information that the operating system supplies. ♦
- The database server generates the second part of the audit record, with fields that depend on the audit event.

IDS

Figure 3-1 on page 3-4 shows the format of the database server audit records for Dynamic Server.

Figure 3-1
Audit-Record Format for Dynamic Server

ONLN	date and time	hostname or hostname. domain.ext	pid	database server name	user name	errno	event mnemonic	Additional Fields
ONLN	1998-07-28 15:43:00.000	turk	4549	khan	jazt	0	CRDB	dbsch
ONLN	1998-07-28 15:43:18.000	turk	4549	khan	jazt	0	ACTB	dbsch:jazt:v1:103
ONLN	1998-07-28 15:43:19.000	turk	4549	khan	jazt	0	CLDB	dbsh
ONLN	1998-07-28 15:43:21.000	turk	4549	khan	jazt	0	ALFR	local:109:-:-:4:4:db1,db2,db3,rootdbs
ONLN	1998-07-28 15:43:28.000	turk	4549	khan	jazt	0	ALFR	local:109:aa5x:-:32:4:db1,db2
ONLN	1998-07-28 15:43:29.000	turk	4549	khan	jazt	0	STDS	2:-
ONLN	1998-07-28 15:43:29.000	turk	4549	khan	jazt	0	STPR	100
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

ONLN — A fixed field used to identify Dynamic Server events

date and time — Indicates when the audit event was recorded

hostname — The name of the UNIX host computer of the client application that executes the audit event

hostname.domain.ext — The name of the Windows host computer, domain, and extension of the client application that executes the audit event

pid — The process ID of the client application that causes the database server to execute the audit event

database server name	The name of the database server on which the audit event is executed
user name	The login name of the user who requests the event
errno	The event result that contains the error number that the event returns, indicating success (0) or failure
event mnemonic	Database server audit event that the database server executed, such as ALFR (Alter Fragment)
additional fields	Any fields that identify databases, tables, and so on. These additional fields are audit-event fields that contain information captured in tabular form by the **onshowaudit** utility for audit analysis. For operating-system-managed auditing on UNIX, the database server audit record is an additional field for the operating-system audit record. Appendix A lists the audit-event fields.

♦

XPS

For Extended Parallel Server, audit records appear in the format shown in Figure 3-2.

Figure 3-2
Audit-Record Format for Extended Parallel Server

XPS	coserver id	datetime	host name	PID	database server name	username	errno	event mnemonic	additional fields
xps	2	2000-01-29 03:23:51.000	smart	1752	dragon	sunny	0	ONST	-
xps	2	2000-01-29 03:23:53.000	smart	1754	dragon	sunny	0	STSN	
xps	2	2000-01-29 03:23:53.000	smart	1754	dragon	sunny	0	OPDB	sysmaster: 0:-

(1 of 2)

XPS	coserver id	datetime	host name	PID	database server name	username	errno	event mnemonic	additional fields
XPS	2	2000-01-29 03:23:53.000	smart	1754	dragon	sunny	0	CLDB	sysmaster
XPS	2	2000-01-29 03:24:44.000	smart	964	dragon	sunny	0	ONST	-
XPS	2	2000-01-29 03:24:46.000	smart	964	dragon	sunny	0	ONST	-d
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

(2 of 2)

The output for Extended Parallel Server includes the following fields that are different from the output for Dynamic Server:

XPS A fixed field that identifies Extended Parallel Server events

coserver id The number of the coserver where the event occurred

♦

XPS

Audit Record Output Sample for Extended Parallel Server

The output sample shown in Figure 3-3 shows the audit trail of events on Extended Parallel Server, one of which requires more than one record to completely describe the event.

Figure 3-3
Example Audit Records for SELECT Statement

xps | 2 | 2000-01-29 03:23:51.000 | smart | 2234 | dragon | sunny | 0:OPDB:sysmaster:0:-
xps | 2 | 2000-01-29 03:24:23.000 | smart | 2234 | dragon | sunny | 0:SLCT:1:1:select * from customers;
xps | 2 | 2000-01-29 03:26:53.000 | smart | 2234 | dragon | sunny | 0:SLCT:1:1:select items.stock_num, items.price, orders.order_num, items.manu_code from items, orders where item.price > 1000 and item.manu_code = 'abcdefghijklmnopqrstuvwxyz123456789012345678901234abcdefghijklmnopqrs
xps | 2 | 2000-01-29 03:26:53.000 | smart | 2234 | dragon | sunny | 0:SLCT:1:2: stuvwxyz123456789012';
xps | 2 | 2000-01-29 03:30:15.000 | smart | 2234 | dragon | sunny | 0:SLCT:0:1:select count(stock_num) from item;

During this audit trail, three cursor SELECT events occurred. The first and third event execute short queries whose contents fit into the additional field. Each of these events generates a single audit record with the **flag** field set to `1`. The second SELECT event, however, executes a much longer query that requires two audit records to include all of its information. The **flag** field in the first record is set to `1`, while the **flag** field in the second record is set to `2`.

Audit Analysis Without SQL

Use the **onshowaudit** utility to extract data for audit analysis. This utility can perform some basic filtering such as user or database server name. You can then send the extracted data to standard output (for example, your screen) and use UNIX utilities such as **grep**, **sed**, and **awk** or Windows utilities to analyze it. You can also choose to put the data in a database and analyze it with SQL, as the next section describes.

Only the AAO can execute **onshowaudit**. If role separation is not enabled, user **informix** will be the AAO. (Superuser **root** on UNIX is always an AAO.) Because disclosure of audit records represents a security threat, only the AAO should read the extracted records.

For example, the following command extracts audit records for the user **pat** from an operating-system-managed audit file named **laurel.12**, on UNIX, and sends the audit records to standard output:

```
onshowaudit -I -f laurel.12 -u pat
```

The command-line syntax for how to extract information with **onshowaudit** is explained in Chapter 4.

Audit Analysis with SQL

You can also use the **onshowaudit** utility to reformat the extracted data and redirect it to a data file and then use the **dbload** utility to load that data into a database table. This section explains this process.

Planning for SQL Audit Analysis

When you plan audit analysis with the database server, consider that the audit-analysis process itself might generate audit records, depending on how the audit is configured. One way to avoid generating unwanted audit records as a result of audit analysis is to use a separate unaudited instance of the database server.

To perform audit analysis with SQL, you must use a program to access the database and table that you created. Use the DB-Access utility to construct and execute SQL statements or develop an application with an IBM Informix application development tool or an SQL API, such as IBM Informix ESQL/C.

Whether you perform analysis with DB-Access or build a customized application, remember the advice given for audit review in "Audit Analysis" on page 1-26. To view audit events for specific objects, select rows based on their value in the **dbname**, **tabid**, or **row_num** column.

If you discover suspicious activity based on initial analysis of the audit table in the database server, you might increase the scope of your collection of audit events to pinpoint the problem. If you feel certain you have a security problem, see "DBMS Security Threats" on page 1-32.

Revoking and Granting Privileges to Protect Audit Data

When you create a database as described in the following sections, make sure that the database is protected against unauthorized access.

Tables that you create in non-ANSI compliant databases have privileges that allow all users access. Although the default database permissions or access privileges prevent access to the tables, proper security practice protects the audit-analysis table in a database that is not ANSI-compliant by revoking access from all other users as soon as that table is created.

You can use the following SQL statements to control access:

```
REVOKE ALL ON table FROM PUBLIC
GRANT ALL ON table TO informix
```

After table privileges are revoked, generally with the REVOKE statement, you can grant individual users (for example, user **informix**) access to the tables with the GRANT statement. For information on SQL statements, see the *IBM Informix Guide to SQL: Syntax*.

Tables created in ANSI-compliant databases have privileges that allow access only by the owner, which is the appropriate security measure.

You can also use the **NODEFDAC** environment variable to control access. When set to yes, **NODEFDAC** does not allow default table privileges (Select, Insert, Update, and Delete) to be granted to PUBLIC when a new table is created in a database that is not ANSI-compliant. For details, see the *IBM Informix Guide to SQL: Reference*.

IDS

Preparing Audit Analysis Records for SQL Access on Dynamic Server

Take the following steps to prepare audit records for SQL analysis:

1. Create a data file to use with **dbload**.
2. Create a database and table in which to store the audit data.
3. Create a command file to use with **dbload**.
4. Load the audit data into the table.

Creating a Data File for dbload

The first step to prepare for SQL-based audit analysis is to use **onshowaudit -l** to extract selected audit records in **dbload** format and put them in an output file. The following example extracts audit records for the user **pat** from the database server-managed audit file **laurel.11** and directs the records to the **records_pat** output file:

```
onshowaudit -I -f laurel.11 -u pat -l > records_pat
```

Important: *You must remove the six header lines that appear in the output file before you use the file as input for the* ***dbload*** *utility because* ***dbload*** *cannot process the header lines.*

The command-line syntax to extract information with **onshowaudit** is explained in Chapter 4.

Creating a Database and Table for Audit Data

To load data files into a database with **dbload**, a database and table to receive the data must already exist. This section explains how to create the necessary database and table.

Creating a Database

Create a database to hold copies of audit records with the CREATE DATABASE statement. By default, the CREATE DATABASE statement creates the database with privileges that allow access only to the owner, which is the appropriate security measure. It is not necessary to use logging within a database created strictly for audit analysis because the data should not be modified.

The following SQL statement creates a database called **auditlogs97**:

```
CREATE DATABASE auditlogs97
```

You can also create an ANSI-compliant database. Although an ANSI-compliant database has the additional overhead of logging, its treatment of table permissions or access privileges makes it attractive in a secure environment. For more information about UNIX permissions or Windows access privileges, refer to "Revoking and Granting Privileges to Protect Audit Data" on page 3-9.

The following SQL statement creates an ANSI-compliant database:

```
CREATE DATABASE auditlogs97 WITH LOG MODE ANSI
```

Creating a Table

Create a table to hold audit data with the CREATE TABLE statement. The order and data types of the columns is important. Use the order shown in the example in Figure 3-4. The sample schema reflects the format of the **dbload** data file that **onshowaudit** created.

The sample CREATE TABLE statement in Figure 3-4 creates an audit table with the name **frag_logs**. For information about the contents of each column, see "Interpreting Data Extracted from Audit Records" on page 3-16.

The sample CREATE TABLE statement in Figure 3-4 does not include the WITH CRCOLS option, which is for conflict resolution during database replication. To replicate the audit database, use WITH CRCOLS in the CREATE TABLE statement.

```
CREATE TABLE frag_logs (
    adttag CHAR(4),
    date_time DATETIME YEAR TO FRACTION(3),
    hostname CHAR(18),
    pid INT,
    server CHAR(18),
    username CHAR(8),
    errno INT,
    code CHAR(4),
    dbname CHAR(18),
    tabid INT,
    objname CHAR(18),
    extra_1 INT,
    partno INT,
    row_num INT,
    login CHAR(8),
    flags INT,
    extra_2 VARCHAR(160,1));
```

Figure 3-4
Sample CREATE TABLE Statement for a Dynamic Server Audit Table

The table that the statement in Figure 3-4 creates does not have any indexes. To improve audit-analysis performance, you can place indexes on columns within the table, depending on the type of analysis that you perform. For guidance on indexing columns, see your *Performance Guide*.

Creating a Command File for dbload

To load the audit information into the table that you created, first create an ASCII command file for the **dbload** utility. This command file must specify the number of columns and the field delimiter that are used in the data file that **onshowaudit** created. For a description of command files and their use with **dbload**, see the *IBM Informix Migration Guide*.

Include the following information when you create the command file for **dbload**:

Delimiter	|
Number of columns	17
Table name	Table you created to receive the data
Data file name	Output file you create (to serve as input for **dbload**)

The following example uses the **FILE** statement to create a command file for **dbload**. The example includes the **records_pat** data file created in "Creating a Data File for dbload" on page 3-10 and the **frag_logs** table created in "Creating a Table" on page 3-11.

```
FILE records_pat DELIMITER '|' 17;
INSERT INTO frag_logs;
```

You now have the tools necessary to load a data file into the table that you created.

Loading Audit Data into a Database

After you have the database, table, data, and command files for audit analysis, you can load the audit data into the table with **dbload**.

The following example executes the commands specified in the **user_records** command file to load data into the **auditlogs97** database created in "Creating a Database" on page 3-10:

```
dbload -d auditlogs97 -c user_records
```

After the data is loaded, begin your audit analysis with SQL.

XPS

Preparing Audit Analysis Records for SQL Access on Extended Parallel Server

Prepare audit analysis records in the following steps:

1. If this is the first time you extract and load audit records, create the database and table into which you load the external table data. Then create an external table definition that has the same column format as the database table.

   For later operations, you can truncate the existing table and reuse it for new data. For other possible ways to reuse or expand the analysis tables, refer to your *Administrator's Reference* and the *IBM Informix Extended Parallel Server Performance Guide*.
2. With **onshowaudit**, extract the audit records you want to analyze into the external table.
3. Enter the SQL statement that loads data from the external table into the database table.

Creating the Database and Tables

Before you can load audit records into a table, you must create a database and a table to hold the records. By default, the CREATE DATABASE statement creates the database with privileges that allow access only to the owner, which is the appropriate security measure. You might create an ANSI-standard database for additional security, as mentioned in "Revoking and Granting Privileges to Protect Audit Data" on page 3-9.

The following SQL statement creates a database called **auditlogs2000**:

```
CREATE DATABASE auditlogs2000
```

After you create the database, create an OPERATIONAL table to contain the data for SQL access. To create a table named **audit_logs** with appropriate column names and data types for extracted audit data, use the statement that appears in Figure 3-5 and add location and fragmentation information as appropriate.

```
CREATE OPERATIONAL TABLE audit_logs (
    adttag CHAR(4),
    cosvr_id INT,
    date_time DATETIME YEAR TO FRACTION(3),
    hostname CHAR(18),
    pid INT,
    server CHAR(18),
    username CHAR(8),
    errno INT,
    code CHAR(4),
    dbname CHAR(18),
    tabid INT,
    objname CHAR(18),
    extra_1 INT,
    partno INT,
    row_num INT,
    login CHAR(8),
    flags INT,
    extra_2 VARCHAR(160,1))
...;
```

Figure 3-5
Sample CREATE TABLE Statement for an XPS Audit Table

You create an OPERATIONAL table so that you can use the Express mode when you load the data from the external table. For information about the characteristics of OPERATIONAL tables, refer to the *IBM Informix Extended Parallel Server Performance Guide*.

After you create the database and table, use the definition of the database table, **audit_logs**, to create an external table definition that is used as a formatting and conversion template when you load the extracted audit data. The external table definition also specifies a file to contain records that are rejected during the load process. Because **onshowaudit** uses a pipe character as the delimiter, you do not need to specify the delimiter character. The statement that defines the external table appears in Figure 3-6.

```
CREATE EXTERNAL TABLE load_audit
SAMEAS audit_logs
(FORMAT "DELIMITED",
   DATAFILES ("disk:1:/data/data2load.tbl.1"
   REJECTFILE "/tmp/load_reject");;
```

Figure 3-6
Sample CREATE EXTERNAL TABLE Statement

After you create the database and internal table and define the external table, you do not need to repeat these steps.

Extract Audit Records to Load into the External Table

Use **onshowaudit -l** to extract selected audit records into an output file. The following example shows how to extract audit records for the user **pat** from all database server-managed audit files and to redirect the records to the **data2load** output file:

```
onshowaudit -I -u pat -l > data2load
```

Important: *Before you use the output file as input to the external file, remove the six header lines.*

For information about the command-line syntax to extract information with **onshowaudit,** see "The onshowaudit Utility" on page 4-15.

Loading Data from the External Table to the Database Table

To use the external table definition to load data from the **data2load** data file into the **audit_logs** database table, execute the SQL statement shown in Figure 3-7.

```
INSERT INTO audit_logs
   SELECT * FROM load_audit;
```

Figure 3-7
Sample INSERT Statement to Load Data

For more information about using external tables to load data, refer to your *Administrator's Reference*.

Interpreting Data Extracted from Audit Records

When you create a database table to contain audit records that you extract from audit files, you provide a column for each field in the audit record. Figure 3-8 lists recommended column names that are used in Figure 3-4 on page 3-11 and Figure 3-5 on page 3-14 and describes the information that each column contains.

Figure 3-8
Audit-Event Columns in Database Table for SQL Access

Column Name	Description
adttag	ONLN or XPS, depending on the database server
cosvr_id	The number of the coserver on which the audited event occurred (XPS only)
date_time	The date and time of the audited event
hostname	The database servername
pid	The process ID
server	The database server name
username	The username associated with the audited event
errno	The error number, if any

(1 of 2)

Column Name	Description
code	The error code, if any
dbname	The name of the database
tabid	The ID number of the affected table
objname	The index name and the table name, or similar identifier (Not in audit tables created with Informix database servers prior to Version 7.0)
extra_1	Information specific to the object and event, as shown in "Audit-Event Fields" on page A-7
partno	Fragmentation information (Not in audit tables created with Informix database servers prior to Version 7.0)
row_num	The physical row number in the affected table, which combines the row ID and the old row ID and identifies each row for the events Read Row (RDRW), Insert Row (INRW), Update Current Row (UPRW), and Delete Row (DLRW)
login	The user login name
flags	The flag set for the event, as shown in "Audit-Event Fields" on page A-7
extra_2	Information determined by the flag. For examples, see "Audit Record Output Sample for Extended Parallel Server" on page 3-7.

(2 of 2)

Chapter

4

Utility Syntax

In This Chapter

This chapter contains syntax and usage information for the following utilities:

- The **onaudit** utility performs the following operations on both UNIX and Windows:
  - Displays audit masks
  - Creates audit masks
  - Modifies audit masks
  - Deletes audit masks
  - Shows the audit configuration
  - Changes global auditing activities
  - Enables and disables auditing
  - Sets the error mode
  - Establishes mandatory auditing for various administrative roles
  - Starts a new audit file in the audit trail
  - Sets the directory in which audit files reside
  - Specifies the maximum size for each audit file
- The **onshowaudit** utility performs the following operations on both UNIX and Windows:
  - Extracts audit information from the audit trail
  - Prepares extracted audit data for the **dbload** utility

UNIX

- The **onaudit** utility also performs the following operations on UNIX:

  Determines whether the database server or the operating system manages the audit trail ♦

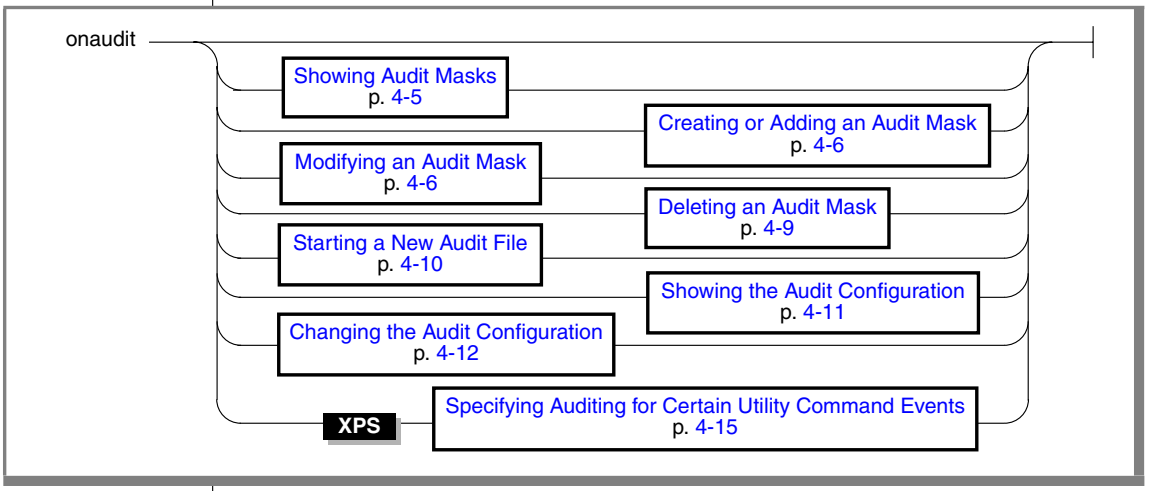
Windows
IDS

- The **onaudit** utility also performs the following operation on Windows:

  Establishes auditing that the database server manages ♦

The onaudit Utility

The **onaudit** utility manages audit masks and auditing configuration.



If your system has role separation, only the DBSSO or AAO can run the **onaudit** utility**.** The DBSSO can perform only **onaudit** functions that involve audit masks, and the AAO can perform only **onaudit** functions that involve audit configuration parameters. Without role separation, the user **informix** or **root** can perform all these tasks.

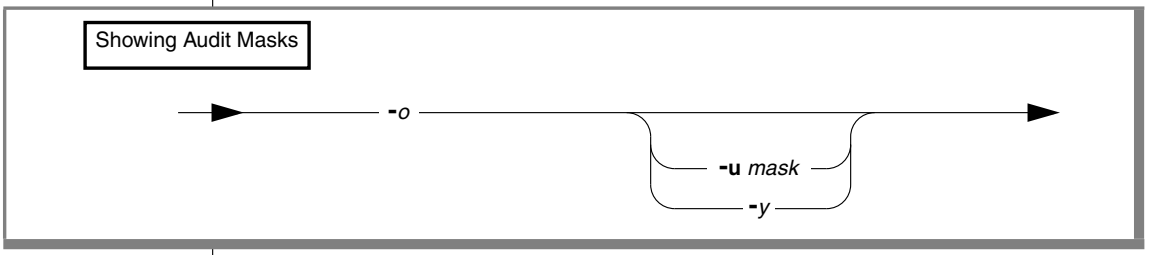
The DBSSO can change audit masks dynamically. Changes to user, default, template, and global masks become effective immediately for user sessions.

XPS

You must run the **onaudit** command from coserver 1. All changes affect all coservers. ♦

If you run the **onaudit** command without any options, it displays a usage summary.

Showing Audit Masks



Element	Purpose	Key Considerations
-o	Outputs audit masks.	None.
-u *mask*	Names a specific mask to display.	Can be any existing audit mask.
-y	Automatically responds `yes` to the confirmation prompt.	None.

The **-o** option of the **onaudit** utility sends the mask display to standard output, as follows:

- If the **-u** *mask* option is omitted, all masks are displayed.
- If the **-y** and **-u** options are omitted, **onaudit** requests confirmation before it displays all the masks, which can amount to a lot of data.

The following example illustrates the format of the output file. The format is the same as that of an input file for **onaudit**, as "Modifying an Audit Mask" on page 4-6 describes.

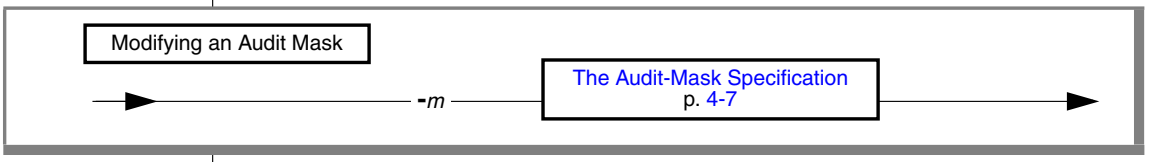
```
maskname basemask audit_events
```

Because the database server keeps no record of the base mask that is used to create or modify a mask, a single hyphen (-) always appears in the *basemask* placeholder.

The following example shows output for the command **onaudit -o -u pat**. It indicates that the individual user mask **pat** contains the Lock Table (LKTB), Create Table (CRTB), and failed attempts of Add Chunk (ADCK) audit events.

```
pat       -      LKTB,CRTB,FADCK
```

Modifying an Audit Mask

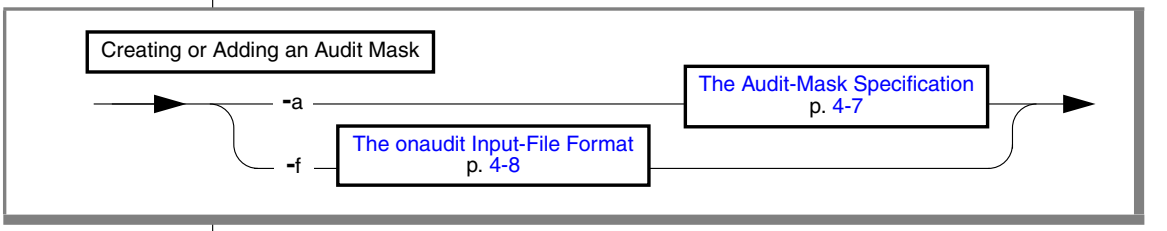


Element	Purpose	Key Considerations
-m	Modifies an existing audit mask.	None.

The following example modifies an audit mask for the user **pat**. The modified mask audits the events specified in the **_Hsecure** template mask, with the addition of all attempts of Lock Table (LKTB) and only failed attempts of Alter Table (ALTB).

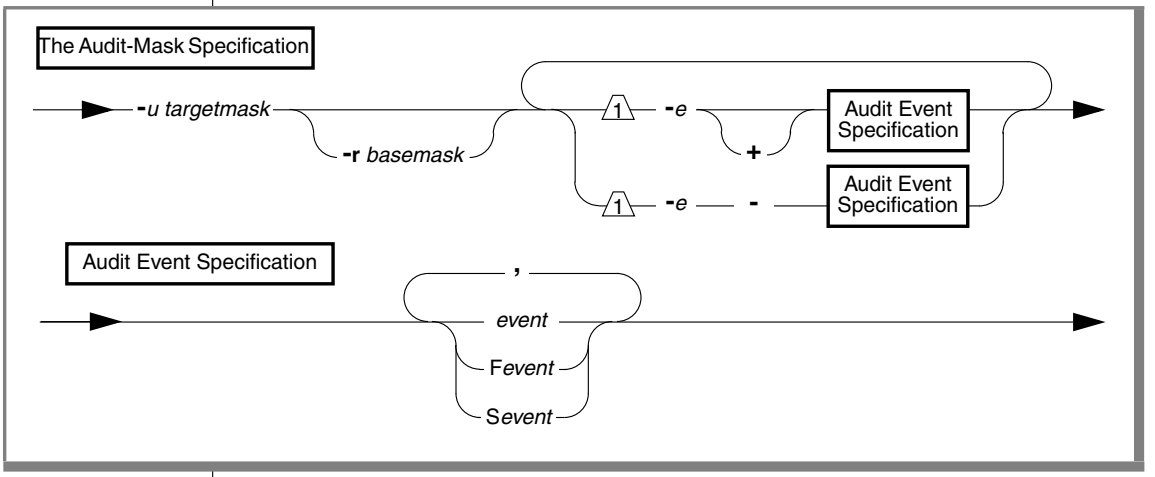
```
onaudit -m -u pat -r _Hsecure -e +LKTB,FALTB
```

Creating or Adding an Audit Mask



Element	Purpose	Key Considerations
-a	Adds a new audit mask.	None.
-f	Names a file that can include instructions to add any or all of the audit masks to the mask table.	References: The syntax for the input file is described in "The onaudit Input-File Format" on page 4-8.

The Audit-Mask Specification



Element	Purpose	Key Considerations
+	Events that follow are to be added to *targetmask* list of audit events	The + is the default and thus is optional.
–	Events that follow are to be dropped from *targetmask* list of audit events	None.
-e	Indicates that audit events are to be added or removed from *targetmask*	Events specified as arguments to **-e** override events listed in any base mask specified with the **-r** option.
-r *basemask*	Name of an existing audit mask. Events currently listed in *basemask* are applied to *targetmask*.	Subsequently changes to *basemask* are not reflected in masks for which *basemask* has been used as a base. If no *basemask* is specified and no events are specified with the **-e** flag, **onaudit** creates an empty target mask.
-u *targetmask*	Names a user, template, **_default**, **_require**, or **_exclude** mask to be created or modified.	'The *targetmask* identifier must have eight or fewer characters.
F*event*	Specifies that only failed event attempts are to be audited.	The *event* can include the event code (mnemonic) for any event listed in the table "Audit-Event Mnemonics for IBM Informix Dynamic Server" on page A-2.
S*event*	Specifies that only successful event attempts are to be audited.	Same as for **F***event*
event	An event to audit, whether the event execution succeeds or fails.	Same as for **F***event*

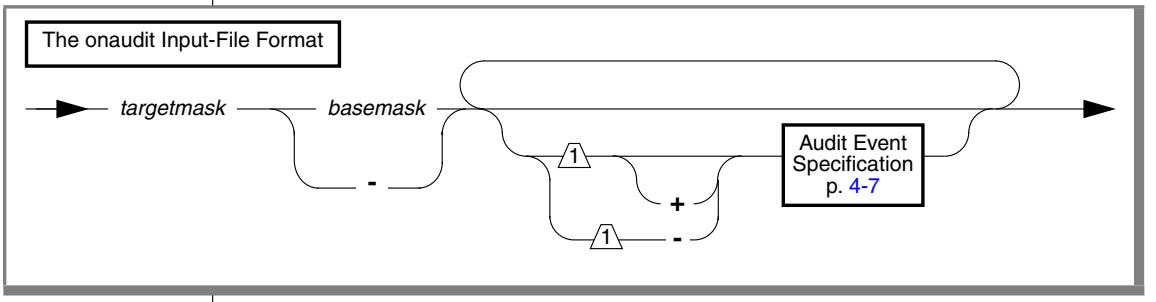
Warning: *Do not include any spaces in the events list. You might get unpredictable results.*

The following example creates a new audit mask named **pat** for the user **pat**. The new mask audits the events specified in the **_secureL** template mask, but excludes Read Row (RDRW) and includes Lock Table (LKTB), successful attempts at Add Chunk (ADCK), and all attempts at Create Table (CRTB).

```
onaudit -a -u pat -r _secureL -e -RDRW, -e +LKTB,SADCK,CRTB
```

A user mask is only one of the three masks that specify auditing for an individual. Auditing instructions are read from the user mask first, followed by the **_require** and **_exclude** masks. For details, refer to Chapter 1.

The onaudit Input-File Format



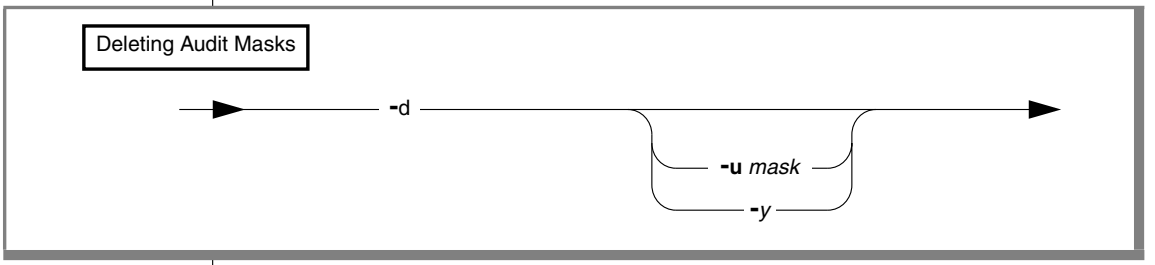
Element	Purpose	Key Considerations
+	Events that follow are to be added to the list of audit events in *targetmask*.	None.
–	Used before an event, it indicates that the events that follow are to be removed from the list of audit events in *targetmask*. Used alone, it creates an empty mask.	None.
basemask	Name of an existing audit mask to use as a base.	The auditing instructions of the base mask are copied to the target mask, in addition to (or except for) the audit events that follow.
targetmask	Identifies the user, template, **_default**, **_require**, or **_exclude** mask to add.	Mask names must not exceed eight characters, and template mask names must begin with an underscore (_) symbol.

The following example uses a modified output file, created by the **onaudit -o** option, as the input file for **onaudit -f**:

```
onaudit -f /work/masks_feb.97
```

For an example of an **onaudit** input file, see Chapter 2.

Deleting an Audit Mask

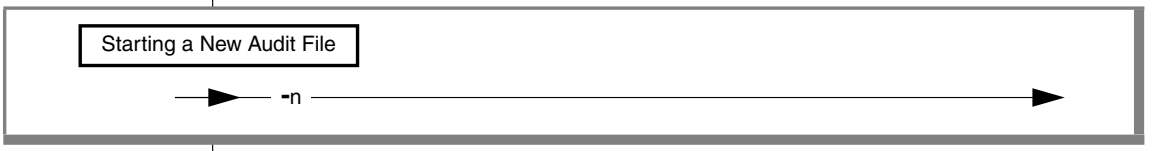


Element	Purpose	Key Considerations
-d	Deletes an audit mask.	None.
-u *mask*	Names a specific mask to delete.	*Mask* can be any existing mask.
-y	Automatically responds `yes` to the confirmation prompt.	None.

The **-d** option of the **onaudit** utility deletes audit masks, as the following list describes:

- If the **-u** *mask* option is omitted, all masks are deleted, including **_default**, **_require**, and **_exclude**.
- Because of the potential to make a significant mistake, the **onaudit** utility prompts you for confirmation before it deletes all masks. Thus, if the **-y** and **-u** options are omitted, **onaudit** requests confirmation.

Starting a New Audit File



Element	Purpose	Key Considerations
-n	Starts a new audit file.	None.

Storing Database Server Audit Files

IDS

For database server-managed auditing, the **-n** option to the **onaudit** utility closes the current database server audit file, stores it in a specified directory, and opens a new audit file named *servername.integer*. The *servername* value is the name of the database server being audited, and *integer* is the next available integer. For example, if the last audit file saved for the **maple** database server was named **maple.123**, the next audit file is saved in a file called **maple.124**. ♦

XPS

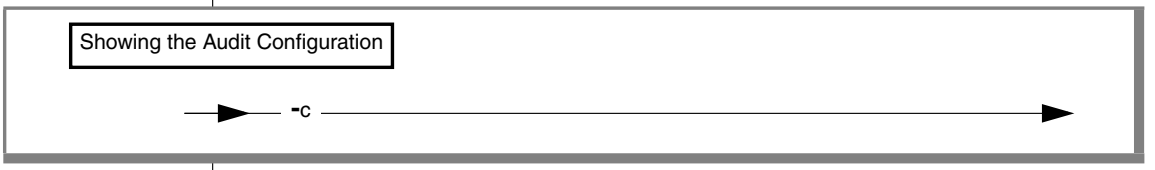
For Extended Parallel Server, the **-n** option to the **onaudit** utility closes the current database server audit file, stores it in the audit-file subdirectory for each coserver, and opens new audit files named *servername.integer* in each coserver-specific subdirectory of the path you specified for audit-file storage.

The subdirectory name is of the form *servername.coserver_id*, which combines the name of the database server and the number of the coserver that hosts the audit file. All audit files are stored locally on the coserver when the audited event occurs. Only one audit file directory exists for each coserver. The names of audit files in this directory are in the same form as for Dynamic Server. For more information, see "Audit File Names" on page 1-20. ♦

Storing Operating-System Audit Files

For operating-system-managed files, the **-n** option to **onaudit** closes the current operating-system audit file, stores it as part of the operating-system audit trail, and opens a new audit file. For the naming conventions for files in the audit trail, see your operating-system documentation.

Showing the Audit Configuration



Element	Purpose	Key Considerations
-c	Shows the current audit configuration.	None.

The **-c** option directs **onaudit** to display the current state of auditing.

UNIX

Figure 4-1 shows an example audit-configuration output on UNIX.

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright (c) IBM Corp., 1998-2003

Current audit system configuration:
      ADTMODE  = 1
      ADTERR   = 0
      ADTPATH  = /tmp
      ADTSIZE  = 20000
      Audit file = 64
```

Figure 4-1 *Sample Audit-Configuration Output on UNIX*

♦

Windows

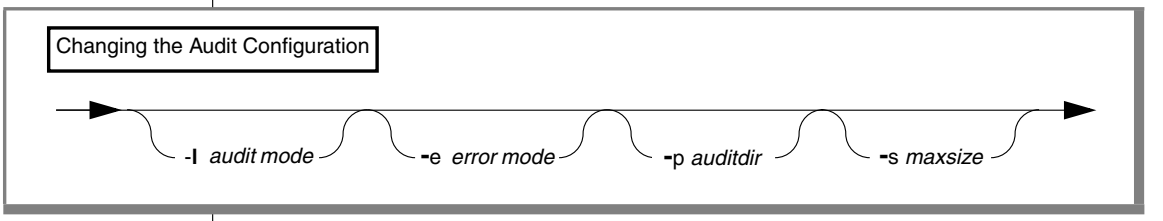
Figure 4-2 shows an example of audit-configuration output on Windows.

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright (c) IBM Corp., 1998-2003

Current audit system configuration:
      ADTMODE  = 1
      ADTERR   = 0
      ADTPATH =    %informixdir%/aaodir
      ADTESIZE =   50000
      Audit file =  0
```

Figure 4-2 *Sample Audit-Configuration Output on Windows*

You can change the audit configuration, as the next section describes. ♦

Changing the Audit Configuration



Element	Purpose	Key Considerations
-e *error mode*	Specifies the error-handling method for auditing when a record cannot be written to the audit file or event log.	**Restrictions**: The *error mode* parameter can have one of the following values: `0`, `1`, `3`. **Additional Information**: This option pertains to the value set for the ADTERR configuration parameter in the ADTCFG file. The value can be changed only when auditing is on. For details of the valid *error mode* values, see "Using the -e Option" on page 4-13.
-l *audit mode*	Specifies the audit mode.	**Restrictions:** The *audit mode* parameter can have one of the following values on UNIX: `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`. The *audit mode* parameter can have one of the following values on Windows: `0`, `1`, `3`, `5`, `7`. **Additional Information**: This option pertains to the value set for the ADTMODE configuration parameter in the ADTCFG file. For details of the valid *audit mode* values, see "Using the -l Option" on page 4-14.

(1 of 2)

Element	Purpose	Key Considerations
-p *auditdir*	Specifies the directory in which the database server creates audit files.	**Restrictions**: You can change the *auditdir* value only for database server-managed auditing and only when auditing is in effect. **Additional Information**: This option pertains to the value set for the ADTPATH configuration parameter in the ADTCFG file. The change occurs with the next write attempt. The database server starts a new audit file in the new directory, beginning with the first available number that is equal to or greater than 0.
-s *maxsize*	Specifies the maximum size (in bytes) of an audit file.	**Restrictions**: The *maxsize* can be any value between 10,240 bytes and approximately 2 gigabytes (the maximum value of a 32-bit integer). If you specify a size that is less than the minimum, it will be set automatically at the minimum. You can specify the *maxsize* value only for database server-managed auditing and only when auditing is in effect. **Additional Information**: This option pertains to the value set for the ADTSIZE configuration parameter in the ADTCFG file. When an audit file reaches or exceeds *maxsize*, the database server closes the current file and starts a new audit file.

(2 of 2)

For information on the audit configuration parameters in the **ADTCFG** file, see Appendix B.

Changes made to the audit configuration with **onaudit** take effect immediately for all user sessions, including existing sessions. For information on how audit-configuration changes interact with the **ADTCFG** file, see Chapter 1.

Using the -e Option

This section discusses the values that you can enter for the **-e** *error mode* option of **onaudit**.

To specify *continue mode*, enter `0` as the argument to the **-e** option. In continue mode, the database server continues processing the thread and notes the error in the message log. Errors for subsequent attempts to write to the audit file are also sent to the message log. For information about the message log, see your *Administrator's Guide*.

To specify one of the *halt mode*s, which suspend processing or shut the database server down, enter one of the following arguments to the **-e** option:

- Enter `1` to suspend processing a thread when the database server cannot write a record to the current audit file and should continue the write attempt until it succeeds.
- Enter `3` to shut down the database server.

Using the -l Option

This section discusses the values that you can enter for the **-l** *audit mode* option of **onaudit**.

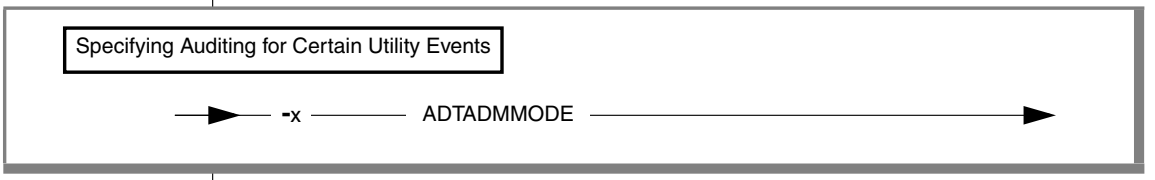
The value `0` turns *off* auditing. The database server stops auditing for all existing sessions, and new sessions are not audited.

The other values all turn *on* auditing, as follows:

- `1` turns on database server-managed auditing for all sessions but does not automatically audit DBSSO and the DBSA actions.
- UNIX On UNIX, `2` turns on operating-system-managed auditing but does not automatically audit DBSSO or DBSA actions. ♦
- `3` turns on database server-managed auditing and automatically audits DBSSO actions.
- UNIX On UNIX, `4` turns on operating-system-managed auditing and automatically audits DBSSO actions. ♦
- `5` turns on database server-managed auditing and automatically audits DBSA actions.
- UNIX On UNIX, `6` turns on operating-system-managed auditing and automatically audits DBSA actions. ♦
- `7` turns on database server-managed auditing and automatically audits DBSSO and DBSA actions.
- UNIX On UNIX, `8` turns on operating-system-managed auditing and automatically audits DBSSO and DBSA actions. ♦

XPS

Specifying Auditing for Certain Utility Command Events



Element	Purpose	Key Considerations
-x	Turns the setting of the ADTADMMODE configuration parameter on and off.	None.

For information about the ADTMMODE configuration parameter, see "ADTADMMODE" on page B-3. ♦

The onshowaudit Utility

The **onshowaudit** utility lets you extract information from an audit trail. You can direct this utility to extract information for a particular user or database server or both. This information enables you to isolate a particular subset of data from a potentially large audit trail.

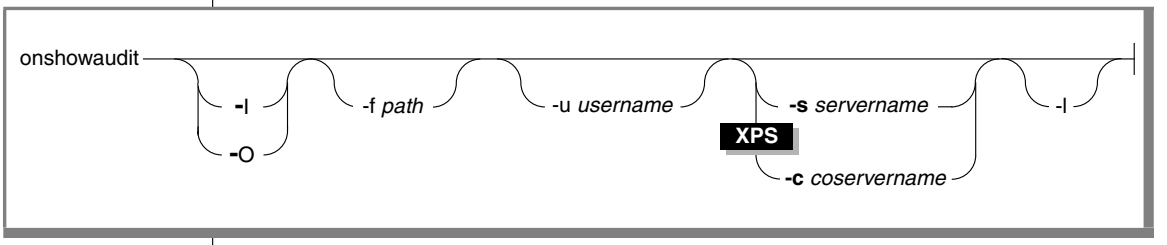
The records are formatted for output. By default, **onshowaudit** displays the extracted information on the screen. You can redirect the formatted output to a file or pipe and can specify that **onshowaudit** reformat the output so you can load it into an Informix database table.

The **onshowaudit** utility extracts data from an audit trail but does not process the records or delete them from the audit trail. Access the audit trail only with the **onshowaudit** utility, which has its own protection:

- With role separation off, only user **informix** (and user **root** on UNIX) can run **onshowaudit**.
- With role separation on, only the AAO can run **onshowaudit**.

UNIX

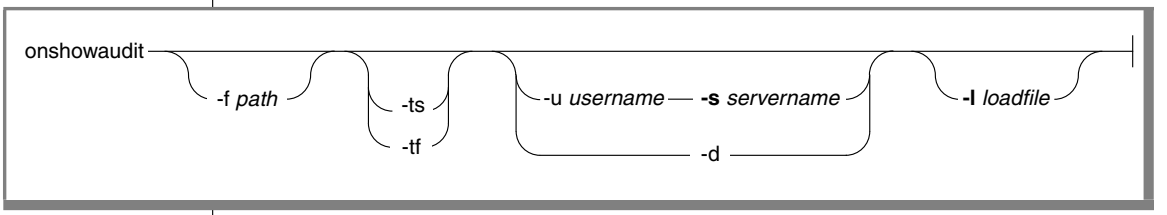
The UNIX command-line syntax for **onshowaudit** follows.



♦

Windows

The Windows command-line syntax for **onshowaudit** follows.



♦

Important: *If you include the* ***-l*** *option in your* ***onshowaudit*** *command, you must remove the six header lines that appear in the output file before you use that file as input for* ***dbload*** *or for an external file.*

The following table identifies the syntax terms that can appear in an **onshowaudit** command line.

Any command-line options that you specify determine which part of the audit trail the **onshowaudit** utility uses.

Element	Purpose	Key Considerations
-d	On Windows, assumes the default values for the user (*current user*) and the database server (**INFORMIXSERVER**)	None
-f *path*	Specifies a specific audit trail to examine, only for database server-managed auditing	If this option is omitted, or if *path* is only a *filename*, see the notes that immediately follow this table.
-I	On UNIX, uses the Informix database server audit trail	None
-l **-l** *loadfile*	Directs **onshowaudit** to extract information with delimiters so that it can be redirected to a file or pipe and loaded into a database table or other application that accepts delimited data	For information on the file format, see Chapter 3. For information on the **dbload** utility, see the *IBM Informix Migration Guide*. For information on loading data with external tables, see your *Administrator's Reference*.
-O	On UNIX, uses the operating-system audit trail	None
-tf	On Windows, shows only *failure* audit records	None
-ts	On Windows, shows only *success* audit records	None
-s *servername*	Specifies the database server about which to extract audit information	None
-c *coserverid*	Specifies the coserver number for which to extract audit information	If omitted, information for all coservers is extracted ♦
-u *username*	Specifies the login name of a user about which to extract audit information	None

XPS

If **-f** is omitted, **onshowaudit** searches for audit files in the **ADTPATH** directory (set with the **onaudit** utility or in the **ADTCFG** file). The **onshowaudit** utility extracts data from all the audit files it finds that are in sequence, starting with the lowest integer.

If an *incomplete pathname* (nothing but a filename) is specified, the **onshowaudit** utility searches the **ADTPATH** directory for that file and extracts audit data from it.

If a *complete pathname* is specified, the **onshowaudit** utility extracts audit data from the named file.

For information on the auditing configuration parameters in the **ADTCFG** file, see Appendix B.

The database server does not audit the execution of the **onshowaudit** utility.

Warning: *Version 7.2 and later versions of the* ***onshowaudit*** *utility can parse and process the new and updated record structures for fragmented tables and indexes, which can span multiple partitions. If you use Version 7.2 or a later version of* ***onshowaudit*** *to analyze records that a database server prior to Version 7.0 created, you might receive inaccurate results. Version 7.2 and later versions of* ***onshowaudit*** *expect to find an additional field for fragmentation (****partno****) in certain audit records, but this field is absent in audit records prior to Version 7.0.*

UNIX

When you use operating-system-managed auditing on UNIX, **onshowaudit** calls operating-system utilities to extract from the operating-system audit trail audit records that the Informix DBMS generates.

Important: *It is recommended that the OSA always enable auditing for utilities that extract audit events from the operating-system audit trail.* ♦

Appendix

A

Audit Events

This appendix contains the following tables:

- Auditable events for each database server, listed alphabetically by event mnemonic

  Refer to "Audit-Event Mnemonics for IBM Informix Dynamic Server" on page A-2 and "Audit-Event Mnemonics for IBM Informix Extended Parallel Server" on page A-5.

- Audit-event records and their fields

  Refer to "Audit-Event Fields" on page A-7.

Important: *The Dynamic Server secure-auditing facility audits only the events that this appendix lists. You might encounter additional SQL statements that the secure-auditing facility does not audit.*

IDS

Audit-Event Mnemonics for IBM Informix Dynamic Server

This table contains an alphabetical list of audit-event mnemonics (event codes) mapped to the name of the event.

Mnemonic	Event Name	Mnemonic	Event Name
ACTB	Access Table	CRDT	Create Distinct Type
ADCK	Add Chunk	CRIX	Create Index
ADLG	Add Transaction Log	CRME	Create Access Method
ALFR	Alter Fragment	CROC	Create Operator Class
ALIX	Alter Index	CROP	Create Optical Cluster
ALME	Alter Access Method	CRRL	Create Role
ALOC	Alter Operator Class	CRRT	Create Named Row Type
ALOP	Alter Optical Cluster	CRSN	Create Synonym
ALTB	Alter Table	CRSP	Create SPL Routine
BGTX	Begin Transaction	CRTB	Create Table
CLDB	Close Database	CRTR	Create Trigger
CMTX	Commit Transaction	CRVW	Create View
CRAG	Create Aggregate	DLRW	Delete Row
CRAM	Create Audit Mask	DNCK	Bring Chunk Off-line
CRBS	Create Storage Space	DNDM	Disable Disk Mirroring
CRBT	Create Opaque Type	DRAG	Drop Aggregate
CRCT	Create Cast	DRAM	Delete Audit Mask
CRDB	Create Database	DRBS	Drop Storage Space
CRDM	Create Domain	DRCK	Drop Chunk
CRDS	Create Dbspace	DRCT	Drop Cast

(1 of 3)

Mnemonic	Event Name	Mnemonic	Event Name
DRDB	Drop Database	LSAM	List Audit Masks
DRDM	Drop Domain	LSDB	List Databases
DRDS	Drop Dbspace	MDLG	Modify Transaction Logging
DRIX	Drop Index	ONAU	**onaudit**
DRLG	Drop Transaction Log	ONBR	**onbar**
DRME	Drop Access Method	ONCH	**oncheck**
DROC	Drop Operator Class	ONIN	**oninit**
DROP	Drop Optical Cluster	ONLG	**onlog**
DRRL	Drop Role	ONLO	**onload**
DRRT	Drop Named Row Type	ONMN	**onmonitor**
DRSN	Drop Synonym	ONMO	**onmode**
DRSP	Drop SPL Routine	ONPA	**onparams**
DRTB	Drop Table	ONPL	**onpload**
DRTR	Drop Trigger	ONSP	**onspaces**
DRTY	Drop Type	ONST	**onstat**
DRVW	Drop View	ONTP	**ontape**
EXSP	Execute SPL Routine	ONUL	**onunload**
GRDB	Grant Database Access	OPDB	Open Database
GRFR	Grant Fragment Access	RDRW	Read Row
GRRL	Grant Role	RLOP	Release Optical Cluster
GRTB	Grant Table Access	RLTX	Rollback Transaction
INRW	Insert Row	RMCK	Clear Mirrored Chunks
LGDB	Change Database Log Mode	RNDB	Rename Database
LKTB	Lock Table	RNTC	Rename Table/Column

(2 of 3)

Mnemonic	Event Name	Mnemonic	Event Name
RSOP	Reserve Optical Cluster	STRL	Set Role
RVDB	Revoke Database Access	STRS	Set Resident
RVFR	Revoke Fragment Access	STRT	Start Statement
RVRL	Revoke Role	STSA	Set Session Authorization
RVTB	Revoke Table Access	STSC	Set Statement Cache
SCSP	SYSTEM Command, SPL Routine	STSN	Start New Session
STCN	Set Constraint	STTX	Set Transaction Mode
STDF	Set Debug File	TMOP	Time Optical Cluster
STDP	Set Database Password	ULTB	Unlock Table
STDS	Set Dataskip	UPAM	Update Audit Mask
STEX	Set Explain	UPCK	Bring Chunk Online
STIL	Set Isolation Level	UPDM	Enable Disk Mirroring
STLM	Set Lock Mode	UPRW	Update Current Row
STOM	Set Object Mode	USSP	Update Statistics, SPL Routine
STOP	Stop Statement	USTB	Update Statistics, Table
STPR	Set Pdqpriority		

(3 of 3)

XPS

Audit-Event Mnemonics for IBM Informix Extended Parallel Server

This table contains an alphabetical list of audit-event mnemonics (event codes) mapped to the name of the event.

Mnemonic	Event Name	Mnemonic	
AASL	Alter Logslice Add Logs	CRLS	Create Logical Logslice
AASS	Alter Dbslice Add Dbspace	CRSL	Create Dbslice
ACTB	Access Table	CRME	Create Access Method
ADCK	Add Chunk	CRSN	Create Synonym
ADLG	Add Transaction Log	CRSP	Create SPL Routine
ALFR	Alter Fragment	CRTB	Create Table
ALME	Alter Access Method	CRTR	Create Trigger
ALTB	Alter Table	CRVW	Create View
BGTX	Begin Transaction	DLRW	Delete Row
CLDB	Close Database	DNCK	Bring Chunk Off-line
CMTX	Commit Transaction	DNDM	Disable Disk Mirroring
CRAM	Create Audit Mask	DPPG	Display Page
CRBT	Create Opaque Type	DRAM	Delete Audit Mask
CRCG	Create Cogroup	DRCG	Drop Cogroup
CRCT	Create Cast	DRCK	Drop Chunk
CRDB	Create Database	DRDB	Drop Database
CRDS	Create Dbspace	DRDS	Drop Dbspace
CRDT	Create Distinct Type	DRIX	Drop Index
CRIX	Create Index	DRLL	Drop Logical Log
CRLL	Create Logical Log	DRLS	Drop Logical Logslice

(1 of 2)

Mnemonic	Event Name	Mnemonic	
DRSL	Drop Dbslice	RLTX	Rollback Transaction
DRSN	Drop Synonym	RVDB	Revoke Database Access
DRSP	Drop SPL Routine	RVTB	Revoke Table Access
DRTB	Drop Table	SCSP	SYSTEM Command, SPL Routine
DRTR	Drop Trigger	SLCT	Select
DRVW	Drop View	STCN	Set Constraint
EXSP	Execute SPL Routine	STDF	Set Debug File
GRDB	Grant Database Access	STDS	Set Dataskip
GRTB	Grant Table Access	STEX	Set Explain
INRW	Insert Row	STIL	Set Isolation Level
LKTB	Lock Table	STLM	Set Lock Mode
LSAM	List Audit Masks	STOP	Stop Statement
LSDB	List Databases	STPR	Set Pdqpriority
MDLG	Modify Transaction Logging	STRT	Start Statement
ONAU	**onaudit**	STSN	Start New Session
ONIN	oninit	STTX	Set Transaction Mode
ONLG	onlog	TCTB	Truncate Table
ONMO	onmode	ULTB	Unlock Table
ONST	onstat	UPAM	Update Audit Mask
OPDB	Open Database	UPCK	Bring Chunk Online
RDRW	Read Row	UPDM	Enable Disk Mirroring
RNDB	Rename Database	UPRW	Update Current Row
RNTC	Rename Table/Column	USTB	Update Statistics, Table

(2 of 2)

Audit-Event Fields

The following table lists audit-event information in alphabetic order by mnemonic code. All events appear in this list, both Dynamic Server events and Extended Parallel Server events.

The list shows the audit-event information that is captured in tabular form by the **onshowaudit** utility for audit analysis:

- The **Event** column shows the event name.
- The **Mnemonic** column lists the acronym that database server utilities use to identify audit events.
- The remaining columns **dbname**, **tabid**, **objname**, **extra_1**, **partno**, **row_num**, **login**, **flags**, and **extra_2** have variable contents, depending on which event a row represents.

  For some events, the **onshowaudit** utility puts two different pieces of information in the **extra_2** field. In this case, the two parts are separated by a semicolon.

Tip: *Granted lists can be long for SQL statements such as GRANT and REVOKE. If the list for an event to be audited does not fit into a single record, the database server creates several audit records to carry the complete information.*

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Alter Logslice, Add Logs	AASL	slicename								
Alter Dbslice Add Dbspace	AASS	slicename							mirror status[1]	
Access Table	ACTB	dbname	owner name, tabid							
Chunk, Add	ADCK	dbspace name			offset				mirror status[1]	path and size
Transaction Log, Add	ADLG	dbspace name			log size					
Alter Fragment	ALFR	dbname	tabid	idxname	operation type[18]			owner	frag flags[15]	dbspaces
Index, Alter	ALIX	dbname	tabid					owner[14]	cluster flag[9, 14]	index name[14]
Access Method, Alter	ALME	dbname	access method ID	access method name				access method owner		
Operator Class, Alter	ALOC									
Optical Cluster, Alter	ALOP	dbname			cluster size			owner		cluster name
Table, Alter	ALTB	dbname	old tabid		new tabid[14]	frag_id				new part-nolist[14]

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Trans-action, Begin	BGTX									
Database, Close	CLDB	dbname								
Trans-action, Commit	CMTX									
Aggregate, Create	CRAG	dbname		aggregate name				owner		
Audit Mask, Create	CRAM							user id		
Storage Space, Create	CRBS	storage space name						owner	mirror status[1]	media
Opaque Type, Create	CRBT	dbname		opaque type name				opaque type owner		
Create Cogroup	CRCG	cogroup name								coserver range
Cast, Create	CRCT	dbname	type ID of from type	function name or "-"	xid of the from type	type ID of the to type	xid of the to type	function owner or "-"		
Database, Create	CRDB	dbspace								dbname
Domain, Create	CRDM									
Dbspace, Create	CRDS	dbspace name							mirror status[1]	

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Distinct Type, Create	CRDT	dbname		distinct type name				distinct type owner		
Index, Create	CRIX	dbname	tabid	idxname				owner	frag flags[15]	dbspacelist
Create Logical Log	CRLL	dbspace name								
Create Logical Logslice	CRLS	logslice name		dbslice name						
Access Method, Create	CRME	dbname	access method ID	access method name				access method owner		
Operator Class, Create	CROC	dbname	operator class ID	operator class name				owner		
Optical Cluster, Create	CROP	dbname	tabid		cluster size			owner		cluster name
Create Role	CRRL	dbname		rolename						
Named Row Type, Create	CRRT	dbname	xid of row type	named row type name				named row type owner		
Create Dbslice	CRSL	dbslice name							mirror status[1]	
Synonym, Create	CRSN	dbname	syn. tabid		base tabid			owner	syn. type[7]	synonym name
SPL Routine, Create	CRSP	dbname	proc. id					owner		procedure name

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Table, Create	CRTB	dbname	tabid	owner				tabname	frag flags[15]	dbspacelist
Trigger, Create	CRTR	dbname	tabid				trigger id[14]	owner[14]		trigger name[14]
View, Create	CRVW	dbname	view tabid					owner		view name
Row, Delete	DLRW	dbname	tabid		partno	frag_id	row-num[14]			
Chunk, Bring Off-line	DNCK				chunk number				mirror status[1]	
Disk Mirroring, Disable	DNDM				dbspace number					
Display Page	DPPG		page-num							
Aggregate, Drop	DRAG	dbname		aggregate name				owner		
Audit Mask, Delete	DRAM							user id		
Storage Space, Drop	DRBS	storage space name								
Cogroup, Drop	DRCG	cogroup name								
Chunk, Drop	DRCK	dbspace name							mirror status[1]	path
Cast, Drop	DRCT	dbname	type ID of from type		xid of the **from** type	type of the **to** type	xid of the **to** type			

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Database, Drop	DRDB	dbname								
Domain, Drop	DRDM									
Dbspace, Drop	DRDS	dbspace name								
Index, Drop	DRIX	dbname	tabid					owner		index name
Transaction Log, Drop	DRLG				log number					
Logical Log, Drop	DRLL		logid							
Logical Logslice, Drop	DRLS	logslice name								
Access Method, Drop	DRME	dbname	access method ID	access method name				access method owner		
Operator Class, Drop	DROC	dbname		operator class name				owner		
Optical Cluster, Drop	DROP	dbname						owner		cluster name
Role, Drop	DRRL	dbname		rolename						
Named Row Type, Drop	DRRT	dbname	xid of dropped type							
Dbslice, Drop	DRSL	dbslice name								
Synonym, Drop	DRSN	dbname	syn. tabid					owner		synname

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
SPL Routine, Drop	DRSP	dbname	proc. id					owner		spname
Table, Drop	DRTB	dbname	tabid	tabname				owner	drop-flags[21]	partnolist
Trigger, Drop	DRTR	dbname					trigger id	owner		trigname
Type, Drop	DRTY	dbname		type name				type owner		
View, Drop	DRVW	dbname	view tabid						drop-flags[21]	
SPL Routine, Execute	EXSP	dbname	proc. id							
Grant Database Access	GRDB	dbname			privilege[5]					grantees[4]
Grant Fragment Access	GRFR	dbname	tabid	fragment	privilege[5, 14]			grantor		grantees[4, 14]
Grant Role	GRRL	dbname		rolename				grantor		grantees
Grant Table Access	GRTB	dbname	tabid		privilege			grantor		grantee[4, 14], update columns, select columns[4, 14]
Row, Insert	INRW	dbname	tabid			frag_id	rowid			
Database Log Mode, Change	LGDB	dbname							log status[6]	
Table, Lock	LKTB	dbname	tabid						lock mode[8]	

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Audit Masks, List	LSAM									
Databases, List	LSDB									
Modify Transaction Logging	MDLG								buffered log flags[2]	
onaudit	ONAU									command line
onbar	ONBAR									command line
oncheck	ONCH									command line
oninit	ONIN									command line
onlog	ONLG									command line
onload	ONLO									command line
onmonitor	ONMN									command line
onmode	ONMO									command line
onparams	ONPA									command line
onpload	ONPL									command line
onspaces	ONSP									command line
onstat	ONST									command line

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
ontape	ONTP									command line
onunload	ONUL									command line
Database, Open	OPDB	dbname							exclu-sive flag	dbpassword
Row, Read	RDRW	dbname	tabid		partno	frag_id	rowid[14]			
Optical Cluster, Release	RLOP	family name					volume number			
Trans-action, Rollback	RLTX									
Chunks, Clear Mirrored	RMCK				dbspace number					
Rename Database	RNDB	dbname		new dbname				user id		
Table/ Column, Rename	RNTC	dbname	tabid	new tab/ colname	colno(*)			owner		tabname(**)
Optical Cluster, Reserve	RSOP	family name					volume number			
Revoke Database Access	RVDB	dbname			privilege[5]					revokees[4]
Revoke Fragment Access	RVFR	dbname	tabid	fragment	privilege[5, 14]			revoker		revokees[4, 14]

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Revoke Role	RVRL	dbname		rolename				revoker		revokees[4]
Revoke Table Access	RVTB	dbname	tabid		privilege[5, 14]			revoker	drop-flags	revokees[4, 14]
SPL Routine, System Command	SCSP									command string
Select	SLCT				type[a]				serial number[b]	query content[c]
Constraint, Set	STCN	dbname							con-straint mode[11]	constraint names
Set Debug File	STDF	dbname								file path
Set Database Password	STDP	dbname						user id		
Set Dataskip	STDS								skip flags[16]	dbspacelist
Set Explain	STEX								explain flags[12]	
Isolation Level, Set	STIL				isolation level					
Set Lock Mode	STLM								wait flags[13]	
Set Object Mode	STOM	dbname	tabid		command mode flag[23]				object typeflag[24]	object names

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Stop Statement	STOP	dbname	tabid							
Set Pdqpriority	STPR								prlevel[17]	
Set Role	STRL	dbname		rolename						
Set Resident	STRS									
Start Statement	STRT	dbname	tabid		Vio_tid				Dia_tid	
Set Session Authorization	STSA	dbname						new username		
Set Statement Cache	STSC			statement name						
Start New Session	STSN									
Set Transaction Mode	STTX				operation[20]				mode flags[19]	
Truncate Table	TCTB	dbname	tabid	tabname						
Optical Cluster, Time	TMOP								time flag[13]	
Table, Unlock	ULTB	dbname	tabid							
Audit Mask, Update	UPAM							user id		

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Chunk, Bring Online	UPCK				chunk number				mirror status[1]	
Disk Mirroring, Enable	UPDM				dbspace number					
Row, Update Current	UPRW	dbname	tabid		old partno		old rowid[14]		new rowid	new partno
SPL Routine, Update Statistics	USSP	dbname	proc. id							
Table, Update Statistics	USTB	dbname	tabid							

(11 of 11)

a. Type: 0 singleton select; 1 cursor select.
For singleton select: a successful event means one row is selected and returned; for cursor select, a successful event indicates a cursor has been opened successfully.

b. The *flags* field records the serial number for the current record within the multiple records created for one single event. These serial numbers range from 1 and up for each set of records and explicitly identify the *order of the audit records that resulted from one single SELECT event. They are useful when you load the audit trail into a database for analysis. See Figure 3-2 on page 3-5 for an example.*

c. The *extra_2* field has a limited size of 160 bytes. The query content can be long, and continuous multiple audit records will be created in the audit trail to carry complete information when necessary.

NOTES

1. Mirror Status: **0** Not mirrored
1 Mirrored

2. Buffered Log Flag: **0** Buffering turned off
1 Buffering turned on

3. Isolation Level:	**0** No transactions **1** Dirty Read **2** Committed Read **3** Cursor Stability **5** Repeatable Read
4. Grantees, Revokees, Select Columns, Update Columns:	These can be lists of comma-separated names. If longer than 166 characters, the audit processing described in "Audit Analysis with SQL" on page 3-8 truncates the lists to 166 characters.
5. Database Privileges:	Table-Level Privileges: **1** Select **2** Insert **4** Delete **8** Update **16** Alter **32** Index **64** Reference **4096** Execute Procedure (When Grant privilege is executed. tabid refers to the procedure ID.) Database-Level Privileges: **256** Connect **512** DBA **1024** Resource
6. Log Status:	**1** Logging on **2** Buffered logging **4** ANSI-compliant
7. Synonym Type:	**0** Private **1** Public
8. Lock Mode:	**0** Exclusive **1** Shared
9. Cluster Flag:	**0** Not cluster **1** Cluster

10. Chunk Flag:
0 Check root reserve size
1 Check entire chunk
<0 Check silently

11. Constraint Mode:
0 Deferred
1 Immediate

12. Explain Flag:
0 Explain turned off
1 Explain turned on

13. Wait Flag:
-1 Wait forever
0 Do not wait
>0 Waiting period (in seconds)

14. If the user request is turned down because of the authorization, those fields are either 0 or blank, depending on the data type.

15. Fragmentation Flag:
0 Not fragmented
1 In dbspace
2 Fragment by round robin
4 Fragment by expression
8 Fragment same as table

16. Skip Flag:
0 DATASKIP for all the dbspaces is turned OFF
1 DATASKIP for the following dbspaces is turned ON
2 DATASKIP for all the dbspaces is turned ON
3 DATASKIP is set to the default

17. Priority Level:
-1 PDQPRIORITY is set to the default
0 PDQPRIORITY is turned OFF
1 PDQPRIORITY is LOW
100 PDQPRIORITY is HIGH
n any other positive integer less than 100 that the user entered in the SET PDQPRIORITY statement

18. Operation Type:
4 Add a new fragment
8 Modify fragmentation
16 Drop a fragment
32 Initialize fragmentation
64 Attach table(s)
128 Detach fragment

19. Mode Flag:
0 Read/Write if operation is Set Access Mode; Dirty Read if operation is Set Isolation Level
1 Read-only if operation is Set Access Mode; Committed Read if operation is Set Isolation Level
2 Cursor Stability
3 Repeatable Read

20. Operation:
0 Set Access Mode
1 Set Isolation Level

21. Dropflags:
0 Cascade
1 Restrict

22. Command Mode Flag:
1 Disabled
2 Filtering without error
4 Filtering with error
8 Enabled

23. Object Type Flag:
1 Constraint
2 Index
3 Constraints and indexes
4 Trigger
5 Triggers and constraints
6 Triggers and indexes
7 All

Appendix

B

The ADTCFG File

This appendix contains a list of the configuration parameters in the **ADTCFG** file and a short discussion of each configuration parameter.

ADTCFG Configuration Parameters

In the discussions in this appendix, each configuration parameter has one or more of the following attributes (depending on their relevance):

default value	Default value that appears in the **adtcfg.std** file
if not present	Value that is supplied if the parameter is missing from your **ADTCFG** file
units	Units in which the parameter is expressed
separators	Separators that can be used when the parameter value has several parts. Do not use white space within a parameter value
range of values	Valid values for this parameter
takes effect	Time at which a change to the value of the parameter actually affects the operation of the database server
utility	Name of the command-line utility that you can use to change the value of the parameter
refer to	Cross-reference to further discussion

ADTCFG File Conventions

The UNIX file **$INFORMIXDIR/aaodir/adtcfg** or the Windows file **%INFORMIXDIR%\aaodir\adtcfg** is called the **ADTCFG** configuration file or simply the **ADTCFG** file. In the **ADTCFG** file, each parameter is on a separate line. The file can also contain blank lines and comment lines that start with a pound (#) symbol. The syntax of a parameter line is as follows:

```
PARAMETER_NAME  parameter_value# comment
```

Parameters and their values in the **ADTCFG** file are case sensitive. The parameter names are always in uppercase letters. You must put white space (tabs, spaces, or both) between the parameter name, parameter value, and optional comment. Do not use any tabs or spaces within a parameter value.

For information about additional Dynamic Server configuration parameters, see the *Administrator's Reference*.

XPS

ADTADMMODE

default value `0`

range of values `0, 1`

0 = ADTADMMODE is off.

ONIN (oninit), ONMO (onmode), and ONST (onstat) events are not audited even if they are specified in a user mask.

`1` = ADTADMMODE is on.

ONIN (oninit), ONMO (onmode), and ONST (onstat) events are audited for all users even if they are not specified in a user mask.

takes effect When **onaudit** is run to change the value or after shared memory is initialized. ADTMODE must be nonzero (auditing is on). See "ADTMODE" on page B-5.

utility **onaudit** (`onaudit -x` *adtadmmode*)

For Extended Parallel Server only, ADTADMMODE specifies whether three of the database server utility-command events are audited. The user mask does not determine whether these events are audited.

ADTERR

default value `0`

range of values `0, 1, 3`

0 = *continue* error mode

When it encounters an error as it writes an audit record, the database server writes a message of the failure into the message log. It continues to process the thread.

`1` = *halt* error mode: suspend thread processing

When the database server encounters an error as it writes an audit record, the database server suspends processing of the thread until it successfully writes a record.

`3` = *halt* error mode: shut down system

When the database server encounters an error as it writes an audit record, the database server shuts down.

takes effect When **onaudit** is run to change the value or after shared memory is initialized. ADTMODE must be nonzero (auditing is on).

utility **onaudit** (`onaudit -e` *errormode*)

ADTERR specifies how the database server behaves when it encounters an error while it writes an audit record.

ADTMODE

default value `0`

range of values `0` through `8`

`0` = auditing disabled

`1` = database server-managed auditing on; starts auditing for all sessions

`2` = operating-system-managed auditing on (UNIX only)

`3` = database server-managed auditing on; audits DBSSO actions

`4` = operating-system-managed auditing on; audits DBSSO actions (UNIX only)

`5` = database server-managed auditing on; audits database server administrator actions

`6` = operating-system-managed auditing on; audits database server administrator actions (UNIX only)

`7` = database server-managed auditing on; audits DBSSO and database server administrator actions

`8` = operating-system-managed auditing on; audits DBSSO and database server administrator actions (UNIX only)

takes effect When **onaudit** is run to change the value or after shared memory is initialized

utility **onaudit** (`onaudit -l` *auditmode*)

ADTMODE controls whether the database server or the operating system manages auditing of user actions on UNIX.

ADTPATH

default value **/tmp** (on UNIX), **%informixdir%\aaodir** (on Windows)

range of values Any valid directory path

takes effect When **onaudit** is run to change the value or after shared memory is initialized

utility **onaudit** (`onaudit -p` *auditdir*)

ADTPATH specifies the directory in which the database server saves audit files. Make sure that the directory that you specify has appropriate access privileges to prevent unauthorized use of audit records.

To change the ADTPATH value with **onaudit**, database server-managed auditing must on.

XPS

For Extended Parallel Server, the path you specify must exist on each node that hosts a coserver. Audit files are stored locally on each coserver, as described in "Location of Audit Files" on page 1-19. The database server creates subdirectories for audit files in the path that you specify. ♦

ADTSIZE

default value 10, 240

units Bytes

range of values Between 10,240 bytes and approximately 2 gigabytes (the maximum value of a 32-bit integer)

takes effect When **onaudit** is run to change the value or after shared memory is initialized

utility **onaudit** (`onaudit -s` *maxsize*)

ADTSIZE specifies the maximum size of an audit file. When a file reaches the maximum size, the database server saves the audit file and creates a new one. This parameter applies only to database server-managed auditing.

Appendix

C

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

> © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. (enter the year or years). All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

AIX; DB2; DB2 Universal Database; Distributed Relational Database Architecture; NUMA-Q; OS/2, OS/390, and OS/400; IBM Informix®; C-ISAM®; Foundation.2000™; IBM Informix® 4GL; IBM Informix® DataBlade® Module; Client SDK™; Cloudscape™; Cloudsync™; IBM Informix® Connect; IBM Informix® Driver for JDBC; Dynamic Connect™; IBM Informix® Dynamic Scalable Architecture™ (DSA); IBM Informix® Dynamic Server™; IBM Informix® Enterprise Gateway Manager (Enterprise Gateway Manager); IBM Informix® Extended Parallel Server™; i.Financial Services™; J/Foundation™; MaxConnect™; Object Translator™; Red Brick Decision Server™; IBM Informix® SE; IBM Informix® SQL; InformiXML™; RedBack®; SystemBuilder™; U2™; UniData®; UniVerse®; wintegrate® are trademarks or registered trademarks of International Business Machines Corporation.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Windows, Windows XP, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names used in this publication may be trademarks or service marks of others.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

W

X

Z