INFORMIX®-Universal Server

Trusted Facility Manual

Table of Contents

Chapter 2   Audit Administration

Introduction

Read this introduction for an overview of the information provided in this manual and for an understanding of the documentation conventions used.

About This Manual

The *INFORMIX-Universal Server Trusted Facility Manual* documents the secure-auditing facility of INFORMIX-Universal Server. This manual provides information on setting up and administrating audit trails, extracting and interpreting audit records, and using 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.

Organization of This Manual

This manual includes the following chapters:

- This Introduction provides an overview of the manual and describes the documentation conventions used.

- Chapter 1, "What Is Auditing?" provides an overview of Universal Server auditing, including audit masks, audit configuration, and audit analysis.

- Chapter 2, "Audit Administration," discusses role separation and the various administrative roles and explains the mechanics of how to set up and administrate an audit trail.

- Chapter 3, "Audit Analysis," explains how to view audit records and how to load them into a database. This chapter also offers some insight into how to interpret audit data.
- Chapter 4, "Utility Syntax," provides detailed descriptions of the **onaudit** and **onshowaudit** Universal Server audit utilities.
- Appendix A, "Audit Events," contains tables of the auditable events with their mnemonics and fields.
- Appendix B, "The ADTCFG File," describes the configuration parameters in the **$INFORMIXDIR/aaodir/adtcfg** file.

Types of Users

This manual is written for system administrators and users of Universal Server who are interested in secure auditing.

It assumes that you are already proficient with computers, understand system administration, and are familiar with the SQL statements pertaining to the events that you want to audit.

If you have limited experience with relational databases, SQL, or your operating system, refer to *Getting Started with INFORMIX-Universal Server* for a list of introductory texts.

Software Dependencies

This manual assumes that you use INFORMIX-Universal Server, Version 9.1, as your database server.

In this manual, all instances of Universal Server refer to INFORMIX-Universal Server.

Assumptions About Your Locale

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

This manual assumes that you are using the default locale, **en_us.8859-1**. This locale supports U.S. English format conventions for dates, times, and currency. In addition, this locale supports the ISO 8859-1 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, 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(s). For instructions on how to specify a nondefault locale, additional syntax, and other consider-ations related to GLS locales, see the *Guide to GLS Functionality*.

Demonstration Database

The DB-Access utility, which is provided with your Informix database server products, includes a demonstration database called **stores7** that contains information about a fictitious wholesale sporting-goods distributor. Sample command files are also included.

Many examples in Informix manuals are based on the **stores7** demonstration database. The **stores7** database is described in detail and its contents are listed in Appendix A of the *Informix Guide to SQL: Reference*.

The script that you use to install the demonstration database is called **dbaccessdemo7** and is located in the **$INFORMIXDIR/bin** directory. For a complete explanation of how to create and populate the demonstration database on your database server, refer to the *DB-Access User Manual*.

Major Features

The Introduction to each Version 9.1 product manual contains a list of major features for that product. The Introduction to each manual in the Version 9.1 *Informix Guide to SQL* series contains a list of new SQL features.

Major features for Version 9.1 Informix products also appear in release notes.

Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other Informix manuals.

The following conventions are covered:

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

Typographical Conventions

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

Convention	Meaning
KEYWORD	All keywords appear in uppercase letters in a serif font.
italics	Within text, new terms and emphasized words appear in italics. Within syntax diagrams, values that you are to specify appear in italics.
boldface	Identifiers (names of classes, objects, constants, events, functions, program variables, forms, labels, and reports), environment variables, database names, filenames, table names, column names, icons, menu items, command names, and other similar terms appear in boldface.
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 feature-, product-, platform-, or compliance-specific information.

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 the documentation, you will find text that is identified by several different types of icons. This section describes these icons.

Comment Icons

Comment icons identify warnings, important notes, or tips. This information is always displayed in italics.

Icon	Description
	The *warning* icon identifies vital instructions, cautions, or critical information.
	The *important* icon identifies significant information about the feature or operation that is described.
	The *tip* icon identifies additional details or shortcuts for the functionality that is described.

Command-Line Conventions

This section defines and illustrates the format of commands that are available in 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.
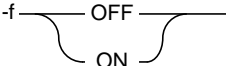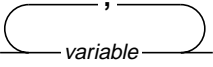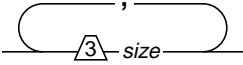
Universal Server supports a variety of command-line options. For example, the **onaudit** and **onshowaudit** utilities described in Chapter 4, "Utility Syntax," require you to issue various commands.

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. Supply a value for words that are 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 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.
-flag	A flag is usually an abbreviation for a function, menu, or option name or for a compiler or preprocessor argument. 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.

(1 of 2)

Element	Description
— ALL —	A shaded option is the default action.
⟶ ⟶	Syntax within a pair of arrows indicates a subdiagram.
⊣	The vertical line terminates the command.
-f — OFF — / ON	A branch below the main path indicates an optional path. (Any term on the main path is required, unless a branch can circumvent it.)
(, variable)	A loop indicates a path that you can repeat. Punctuation along the top of the loop indicates the separator symbol for list items.
(, /3\ size)	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. Here 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 command-line diagram that uses some of the elements that are listed in the previous table.

Figure 1
Example of a Command-Line Diagram

setenv ———— INFORMIXC ———————— compiler ———————— ⊣
                                 pathname

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

diagrams the following steps:

1. Type the word `setenv`.
2. Type the word `INFORMIXC`.
3. Supply either a compiler name or pathname.

    After you choose *compiler* or *pathname*, you come to the terminator. Your command is complete.
4. Press `RETURN` to execute the command.

Sample-Code Conventions

Examples of SQL code occur throughout this manual. Except where noted, the code is not specific to any single Informix application development tool. If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores7
.
.
.
DELETE FROM customer
    WHERE customer_num = 121
.
.
.
COMMIT WORK
DISCONNECT CURRENT
```

To use this SQL code for a specific product, apply the syntax rules for that product. For example, if you use the Query-language option of DB-Access, you must delimit multiple statements with semicolons. If you use an SQL API, you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement.

Tip: Ellipsis points in a code example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept that is discussed.

For detailed directions on how to use SQL statements for a particular application development tool or SQL API, see the manual for your product.

Additional Documentation

For additional information, you might want to refer to the following types of documentation:

- On-line manuals
- Printed manuals
- Error message files
- Documentation notes, release notes, and machine notes
- Related reading

On-Line Manuals

A CD that contains Informix manuals in electronic format is provided with your Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print on-line manuals, see either the installation guide for your product or the installation insert that accompanies the documentation CD.

The documentation set that is provided on the CD describes Universal Server, its implementation of SQL, and its associated application-programming interfaces. For an overview of the manuals in the Universal Server documentation set, see *Getting Started with INFORMIX-Universal Server*.

Printed Manuals

The Universal Server documentation set describes Universal Server, its implementation of SQL, and its associated application-programming interfaces. For an overview of the manuals in the Universal Server documentation set, see *Getting Started with INFORMIX-Universal Server*.

To order printed manuals, call 1-800-331-1763 or send email to moreinfo@informix.com.

Please provide the following information:

- The documentation that you need
- The quantity that you need
- Your name, address, and telephone number

Error Message Files

Informix software products provide ASCII files that contain all the Informix error messages and their corrective actions. To read the error messages in the ASCII file, Informix provides scripts that let you display error messages on the screen (**finderr**) or print formatted error messages (**rofferr**). For a detailed description of these scripts, see the Introduction to the *Informix Error Messages* manual.

Documentation Notes, Release Notes, Machine Notes

In addition to printed documentation, the following on-line files, located in the **$INFORMIXDIR/release/en_us/0333** directory, supplement the information in this manual.

On-Line File	Purpose
TFMANDOC_9.1	The documentation-notes file describes features that are not covered in this manual or that have been modified since publication.
SERVERS_9.1	The release-notes file describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.
IUNIVERSAL_9.1	The machine-notes file describes any special actions that are required to configure and use Informix products on your computer. The file listed contains the machine notes for Universal Server.

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

Related Reading

The auditing facility provided with Universal Server is designed to meet the C2 class of trust as specified in *Trusted Computer System Evaluation Criteria* (CSC-STD-001-83; also known as the "orange book" because of its orange cover) and *Trusted Database Interpretation* (NCSC-TG-021, also known as the "lavender book"). These books are published by the U.S. Department of Defense.

Auditing is only part of the process when you maintain the security of a system. The following publications provide additional information about security. The first publication describes how to maintain a secure system, according to the U.S. Department of Defense *Trusted Computer System Evaluation Criteria*; the second publication discusses issues involved in how to implement and evaluate audit mechanisms.

- *A Guide to Understanding Trusted Facility Management* by the National Computer Security Center (NCSC-TG-015, October 1989)
- *A Guide to Understanding Audit in Trusted Systems* by the National Computer Security Center (NCSC-TG-001, June 1988)

Compliance with Industry Standards

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. 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, on INFORMIX-Universal Server. In addition, many features of Universal Server comply with the SQL-92 Intermediate and Full Level and X/Open SQL CAE (common applications environment) standards.

Informix Welcomes Your Comments

Please tell us what you like or dislike about our manuals. To help us with future versions of our manuals, we want to know about corrections or clarifications that you would find useful. Include the following information:

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

Write to us at the following address:

> Informix Software, Inc.
> SCT Technical Publications Department
> 4100 Bohannon Drive
> Menlo Park, CA 94025

If you prefer to send email, our address is:

> doc@informix.com

Or send a facsimile to the Informix Technical Publications Department at:

> 415-926-6571

We appreciate your feedback.

What Is Auditing?

This chapter provides an overview of INFORMIX-Universal Server auditing and auditing terminology. It describes audit events, explains in detail how audit masks are configured and used, and indicates how to perform audit analysis. It also discusses the various roles involved in auditing.

The INFORMIX-Universal Server Secure-Auditing Facility

Auditing creates a record of selected user activities that are performed. You can use these records for the following purposes:

- To detect unusual or suspicious user actions and identify the specific user(s) who perform 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: *Users should be made aware that every action they take can be audited and that they can be held responsible for those actions.*

Auditing is not a mechanism for keeping track of transactions in order to reconstruct a database. Universal Server has archiving and backup facilities for that purpose, which are explained in the *INFORMIX-Universal Server Backup and Restore Guide.*

The auditing facility provided with Universal Server is designed to meet the C2 class of trust as specified in the *Trusted Computer System Evaluation Criteria* (CSC-STD-001-83; also known as the *orange book* because of its orange cover) and the *Trusted Database Interpretation* (NCSC-TG-021, also known as the *lavender book*), both published by the U.S. Department of Defense.

What Are Audit Events?

Any Universal Server activity that could potentially alter or reveal data or the auditing configuration is considered an *event*. The Universal 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 *mnemonic*, or four-letter abbreviation. The events that you can audit with Universal Server secure-auditing facility are listed and described, and their mnemonics are listed, in Appendix A, "Audit Events."

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

What Are Audit Masks?

Audit masks specify those events that Universal Server should audit. You can include any event in a mask. The masks are associated with user ids, so that specified actions taken by a user id 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. In other words, you need to add events to the masks. (You also need to perform other tasks, which are described in Chapter 2, "Audit Administration.")

Universal Server has no provisions for auditing based on objects or processes. (For example, you cannot ask Universal 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, as explained in Chapter 3, "Audit Analysis."

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.
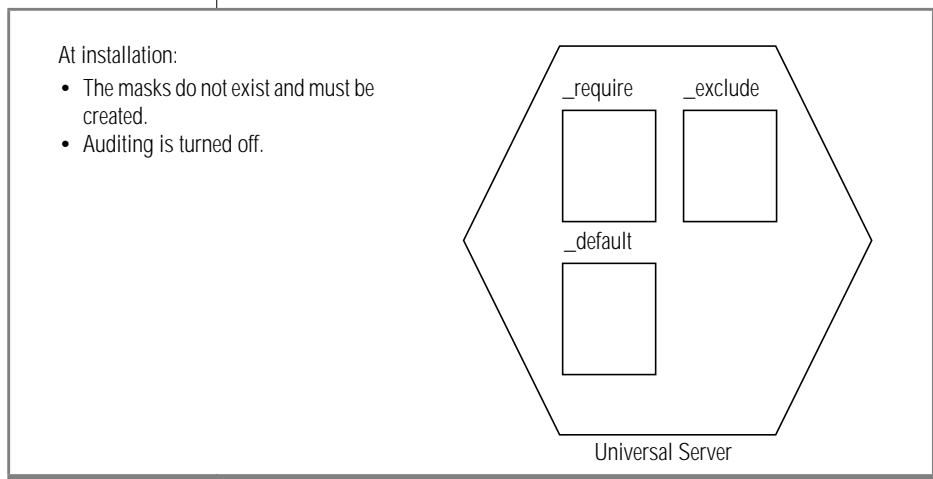
At installation:
- The masks do not exist and must be created.
- Auditing is turned off.

_require   _exclude

_default

Universal Server

Once installation completes, you can create the audit masks and turn auditing on.

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

In addition to the three masks shown in Figure 1-1, you can specify *user masks* for individual users. User masks allow you to audit some users more than others and target different types of activities for different users. A user (other than the audit administrator) cannot tell which events are being audited. User masks are described on page 1-9.

You can also create *template masks*. These masks are used as a template to create new user masks. Template masks are described on page 1-10.

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

Figure 1-2
The Auditing Instructions

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.

Auditing instructions

Global masks

_require   _exclude

_default   user masks

Universal Server
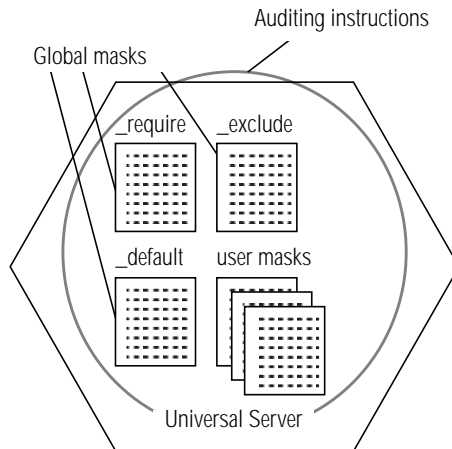
Once you define the auditing instructions and turn on auditing, you can modify one or more audit masks as needs change and potential security threats are identified. See Chapter 2, "Audit Administration," for information on how to change audit masks.

The Auditing Process

When you turn on auditing, Universal Server generates *audit records* for every event the auditing instructions specify, as shown in Figure 1-3. Specify whether the audit records are managed by the operating system or Universal Server. For details, see "Types of Auditing" on page 1-17.

If you use Universal Server-managed auditing, Universal Server stores the audit records in a file called an *audit file*, as shown below. 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, the records are stored in an operating-system audit trail.

Figure 1-3
The Auditing Process
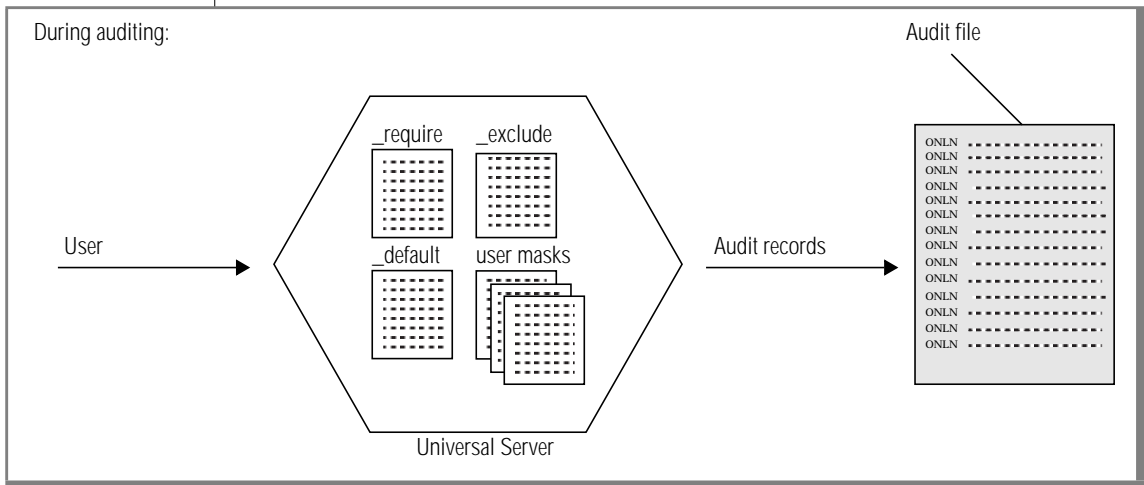
The audit administrator needs to specify and maintain the *audit configuration*, which includes the following information:

- The audit mode
- The directory in which the audit trail is located
- The maximum size of an audit file before automatically starting another audit file
- How Universal Server behaves if it encounters an error when writing audit records to the audit trail

Each of these topics is explained in "Audit Configuration" on page 1-16.

Universal Server generates audit records (and sends them to the audit file) regardless of whether the client that performs the audited action is local or remote. Universal Server includes both the user login and database server name in every audit record to help pinpoint a specific initiator and action.

Reviewing the Audit Trail

Review the audit trail regularly. Universal Server offers a data-extraction utility that you can use to select audit data for specific users or database servers. Once data is extracted, you can specify that it be formatted to load into a database for subsequent manipulation with SQL. This process is explained in "Audit Analysis" on page 1-23.

Administrative Roles

You can set up three types of administrative roles for Universal Server, in addition to any administrative roles your operating system might have:

- The Universal Server administrator maintains and tunes Universal Server.
- The Database System Security Officer (DBSSO) specifies and maintains the audit masks.
- The Audit Analysis Officer (AAO) turns auditing on and off, sets up and maintains the audit configuration, and reads and analyzes audit-trail data.

These roles are optional. The operating-system administrator (OSA), or whoever installs Universal Server, decides at installation time whether to have separate DBSSO and/or AAO administrative roles and who should perform them.

The OSA should set the environment variable **INF_ROLE_SEP** to any value at installation time to enable role separation. If **INF_ROLE_SEP** exists, role separation is enabled. When role separation is not enabled, user **informix** can perform all administrative tasks, and no special groups are needed.

For information on roles and role separation, see Chapter 2.

Audit Masks and Audit Instructions

As described in "What Are Audit Masks?" on page 1-4, an *audit mask* specifies a set of events to be audited when a user performs them. Audited events are derived from a combination of user and compulsory masks. The events are listed in Appendix A. The set of events is fixed, but you can select only the ones that you need.

The four types of audit masks are listed in the following table.

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

The first three kinds of masks are described below in "User Masks". Template masks are described on page 1-10.

User Masks

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

1. The **_default** mask *or* an individual user mask
2. The **_require** mask
3. The **_exclude** mask

When a user initiates access to a database, Universal Server checks whether an individual user mask exists with the same *username* as the account that is used by the user. If an individual mask exists, the audit instructions in it are read first, and the **_default** mask is ignored. If no individual user mask exists, Universal Server reads and applies the audit instructions in the **_default** mask to that user.

In addition to default and individual masks, Universal Server reads and applies the audit instructions in the **_require** and **_exclude** masks. These masks are *compulsory* 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 they are specifically required by the previously read masks.

*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: **user / _default**, **_require**, **_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 the auditing administrator turns on auditing, it operates automatically and cannot be disabled by users.

When Universal Server is installed, no audit masks exist. Even the default mask and the compulsory masks must be specified.

Making a Mask for User informix

*Important: Actions that the Universal Server or audit administrator or user **informix** generally perform are potentially dangerous to the security of the database server. To reduce the risk of an unscrupulous user abusing the **informix** account, Informix recommends 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 (_). This distinguishes them from individual user masks.*

You cannot create template masks with the following names because they are already used by Universal Server:

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

When Universal Server is installed, no template masks exist. The number of template masks you can create is unlimited.

Setting the Audit Instructions

The audit administrator sets the audit instructions that Universal Server performs. This person 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 Universal Server administrator 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 gives some advice on how to choose events to audit and how to set the audit instructions. It also indicates how the choices affect performance. For instructions on how to create and modify audit masks, see Chapter 2, "Audit Administration."

All the audit masks used by Universal Server are stored in the **sysaudit** table in the **sysmaster** database. The masks are updated automatically when Universal 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 discussion of the **sysmaster** database in the *INFORMIX-Universal Server Administrator's Guide*.

Resource and Performance Implications

The amount of Universal Server auditing enabled at any given time has a direct effect on operating-system resources and database server performance. All audit data generated by Universal Server is stored on disk. The greater the number of audit records generated, the greater the amount of disk space required (for storage), and the greater the amount of CPU time required to process audit records (for storage, viewing, deletion, archival, and restoration).

How system resources and performance are affected depends on the following factors:

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

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. On the other hand, 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 Universal Server security-related events that are performed by all users. This action 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 Universal Server user activity). On the other hand, it provides the most audit information, thereby reducing the security risk.

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

The Universal Server event that, if audited, has the most significant effect on system performance and disk space is Read Row (RDRW). Within 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 activity could quickly produce large numbers of audit records.

Suggested Minimum Set of Events to Audit

Although Universal 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.

Informix recommends that you audit the following events for all standard Universal Server users, at all times, with the **_require** audit mask:

- Create Role (CRRL)
- Set Role (STRL)
- Set Session Authorization (STSA)
- Set Object Mode (STOM)
- Open Database (OPDB)
- Grant/Revoke Database Access (GRDB), (RVDB)
- Grant/Revoke Table Access (GRTB), (RVTB)
- 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. It 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 Universal Server users. Auditing all Open operations indicates the general area within Universal Server where users are looking. Auditing these operations should not significantly affect database server performance; these operations are performed infrequently compared with other operations.

Creative attempts to circumvent the Universal Server security policy are virtually impossible to detect if minimal or no auditing is performed for regular Universal Server users. If a security violation is suspected, or if a particular user exhibits unusual behavior (as revealed by the Universal Server audit records in the audit trail), you are advised to enable full auditing for that user. In this way, you can obtain a more complete picture of the activities of that 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 operating-system administrator should make sure that the actions taken during installation are audited.

Level of Auditing Granularity

Universal Server secure-auditing facility can audit the following events at the fragment level of granularity, showing 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.

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

In addition, Universal Server audits the following events to the RESTRICT/CASCADE level:

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

See the *Informix Guide to SQL: Syntax* for more information on the corresponding SQL statements.

Using the Various Masks

The **_require** mask can be a valuable tool; every user of the database server is audited by the events that are specified in this mask. You can use this mask to perform the bulk of the auditing. The **_require** mask allows you to make rapid changes to the auditing configuration 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 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.

The way you use the **_default** and individual user masks depends on the number of users and their activity. For example, if you have only a few users, you might want to give each one an individual mask. The **_default** mask would then be used to audit events initiated by users who do not normally use your database, so you could configure the **_default** mask with a high level of security. You can offset any detrimental effects on system performance if you 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, you can leave the **_default** mask empty and rely on the **_require** mask for most of your auditing.

Audit Configuration

The AAO can monitor the auditing configuration, as described in Chapter 2. The audit configuration consists of the following topics:

- Turning auditing on or off
- Specifying Universal Server-managed auditing or operating-system-managed auditing
- Determining properties of the audit files
- Specifying auditing error modes
- Using the **ADTCFG** file

These topics are described in the following sections.

Turning Auditing On or Off

The audit administrator determines whether auditing is on or off. Auditing is turned off by default when Universal Server is installed.

As described in Chapter 2, the AAO can turn auditing on and off at any time. Universal Server can be in either on-line or quiescent mode for the changes to take effect. The AAO uses the **onaudit** utility, described in Chapter 4, to turn auditing on or off.

When you turn auditing on, all sessions are affected immediately. All user sessions that are started thereafter produce audit records.

Turning auditing off stops auditing for all existing sessions, and new sessions are not audited. If you turn auditing off and then turn it on again while Universal Server is in on-line mode, existing sessions resume producing audit records.

Types of Auditing

When you turn on auditing, you can specify that either Universal Server or your operating system manage audit records. Set the ADTMODE parameter to a number from 0 through 8 in the **ADTCFG** file to determine the type and level of auditing. For details, see "Changing the Auditing Configuration" on page 4-14 and Appendix B, "The ADTCFG File."

For example, if you set the ADTMODE configuration parameter to 1 in your **ADTCFG** file, Universal Server-managed auditing is turned on automatically when Universal Server initializes shared memory. Once auditing is turned on, only the audit events defined in audit masks are recorded. (Note that, if you specify mandatory auditing for the DBSSO and/or Universal Server administrator when you turn on auditing, audit records are generated for all events that are executed by the specified roles.)

The Audit Analysis Officer (AAO) configures auditing so that either Universal Server or the operating system manages the audit records and specifies one of four error modes, should an error occur 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 Operating System Administrator (OSA) administrates 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. The following criteria must be met for Universal Server to use an audit trail managed by the operating system:

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

If audit records are managed by the operating-system audit facility, the amount of Universal Server auditing must also be acceptable to the operating-system administrator.

For more information on auditing administration, see "Administrative Roles and Role Separation" on page 2-3.

Properties of the Audit Files

As described in "The Auditing Process" on page 1-7, with Universal Server-managed auditing, Universal Server writes audit records to audit files in an audit trail. This section describes the audit files in more detail.

The Location of the Audit Files

The audit files are located in a directory that you specify with the **onaudit** utility or the ADTPATH parameter in the **$INFORMIXDIR/aaodir/adtcfg** file, as discussed in Chapter 2, "Audit Administration."

If you change the audit path, the change takes effect immediately for all existing sessions. This feature allows you to change the directory when Universal Server is in on-line 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.

Starting New Audit Files

When an audit record is written, Universal Server appends it to the current audit file. If you bring Universal Server out of on-line mode and then put it back, Universal Server continues to use the same audit file. Universal Server starts a new audit file only under the following conditions:

- When the file reaches a specified size
- When you manually direct Universal Server to start a new audit file, as described in Chapter 2
- If you start Universal Server-managed auditing

Universal 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 value at any time, even when Universal Server writes to an audit file, as described in Chapter 4, "Utility Syntax."

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.

Naming Audit Files

No matter how a new audit file is started, it follows the naming convention *dbservername.integer*, where *dbservername* is the database server name as defined in the ONCONFIG file, and *integer* is the next integer (starting 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.) Note that the names are unique to a specific audit directory, so you can have **auditdir1/maple.123** and **auditdir2/maple.123**, and so on.

Controlling Access to Audit Files

Standard users should not be able to view or alter audit files. The following characteristics control access to audit files and protect them from being accidentally read or destroyed:

Ownership: **root**
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 audit files should be accessed only with the **onshowaudit** utility, which has its own protection.

- With role separation off, only a user who is a member of the group **informix** can run **onshowaudit**.
- With role separation on, only by an AAO who can run **onshowaudit**.

Error Modes When Writing to an Audit File

If Universal 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 described in Chapter 2 at any time during Universal Server operation—even when an error has occurred. See the discussion of **onaudit** error modes in Chapter 4.

Halt Error Modes

When Universal Server is in a *halt* error mode (1, 2, or 3), it does not allow the session that received the error when it writes to the audit trail to continue processing. Universal Server may even terminate the session or shut down, depending on the error mode.

- Mode 1: A thread is suspended but the session continues when the audit record is successfully written.
- Mode 2: The user session is terminated and cannot continue.
- 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. You might need to change the audit directory or start a new audit file.

Continue Error Mode

When Universal Server is in *continue* error mode (0), it allows the session that received the error when it writes to the audit trail to continue processing. (However, the audit record that was being written when the error occurred will be lost.) Universal 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 become huge.

Audit Configuration and the ADTCFG File

The properties of the audit configuration are represented and recorded by parameters in the **$INFORMIXDIR/aaodir/adtcfg** file, also known as the **ADTCFG** file. These parameters are ADTERR, ADTMODE, ADTPATH, and ADTSIZE.

You can edit the **ADTCFG** file to change the audit parameters, but the audit configuration is not changed until you reinitialize shared memory. If you use the **onaudit** utility to change the audit configuration, as described in Chapter 2, 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 is described in the *INFORMIX-Universal Server Administrator's Guide*). The 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 Universal Server administrator to start an instance of Universal Server with invalid audit parameters. See Chapter 4 for details of how to use the **onaudit** utility to configure the **ADTCFG** file.

Controlling Access to Audit-Configuration Files

Standard users should not be able to view or alter the audit trail. The following examples show how to control access to audit files and protect them from being accidentally read or destroyed by users.

The following examples show the security configuration with no role separation:

aaodir

Ownership:	**informix**
Group ID:	**informix**
Permissions:	**774**

aaodir/adtcfg.std

Ownership:	**informix**
Group ID:	**informix**
Permissions:	**664**

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

The following examples show the security configuration with role separation:

aaodir

Ownership:	**informix**
Group ID:	**<aao_group>**
Permissions:	**770**

aaodir/adtcfg.std

Ownership:	**$AAOOWNER**
Group ID:	**<aao_group>**
Permissions:	**664**

Audit Analysis

Audit analysis is performed by the AAO. This section explains the importance of audit analysis, how to prepare for audit analysis, some strategies for audit analysis, and how to react to a perceived security problem.

The Importance of Audit Analysis

The Universal 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. If the audit data is never reviewed or analyzed, this is equivalent to disabling auditing altogether (and is, in fact, worse because it might reduce database server performance).

If, on the other hand, you routinely analyze and review the audit data, suspicious activity might be discovered and acted on before a successful violation. 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: You can play the greatest role in the security of your database server by watching 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 appear 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.

Preparing for Audit Analysis

This section describes two methods to analyze Universal Server audit records.

- The first method is to simply 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 is managed by Universal Server. 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. However, be sure that records are not deleted or modified from either the intermediate file or 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.

Each method relies on a utility called **onshowaudit**, which is discussed in Chapter 3 and described in Chapter 4. For either method, you can extract audit events for specific users and/or database servers.

The preparation process for both analysis methods is shown in Figure 1-4 on page 1-25. Each step is explained in detail in Chapter 3

Figure 1-4
*Preparing for Audit
Analysis*

onshowaudit can extract audit records and display them on your screen (or send them to a pipe or file).

onshowaudit can extract audit records, process them, and create a file for **dbload** to use as an input data file.

Audit records in the audit file

command file

dbload loads the processed records into a database and table with the instructions in a command file.

table

database
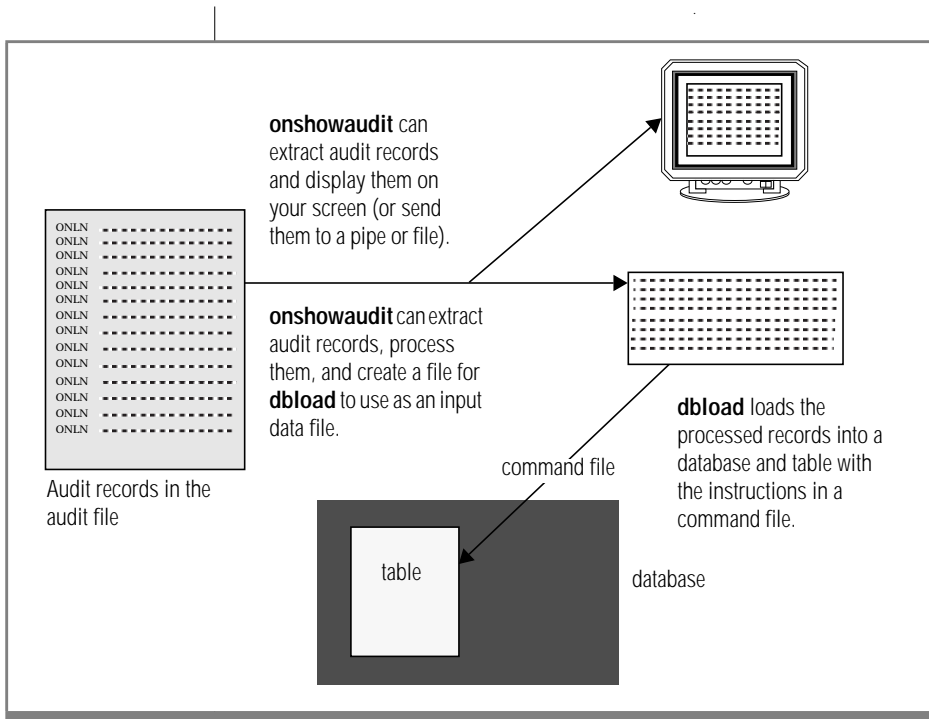
To perform audit analysis, first have audit records in your Universal Server or operating-system audit trail. When the following conditions are present, records are in the operating-system audit trail:

- The operating system supports auditing.

- Universal 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 described in the Machine Notes file. (See "Documentation Notes, Release Notes, Machine Notes" on page 13 of the Introduction.)

- Database server users have performed Universal Server activities that generated audit records.

- Operating-system auditing is on.

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.

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 Universal 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 permission 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 Universal 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 were accessed by this user) 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

A user who searches through stored data to locate or acquire information without a legitimate need is *browsing*. A browser does not necessarily know of the existence or format of the information for which he or she is looking. A browser usually executes 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 the browser 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.

Responding to Identified Security Problems

Once 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 Chapter 2. For a discussion of Discretionary Access Control (DAC) restrictions, see the INFORMIX-OnLine/Secure documentation.

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, Universal Server administrator, 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. This might include:

- a valid login 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
- Stored-procedure level
- Role level
- Fragmentation level

These countermeasures are adequate for legitimate use of the product when users attempt to access the data directly. However, they cannot 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 access rights.

Privileged Activity Threats

Improper or unchecked activity by users with privileged roles (DBSSO, AAO, Universal Server administrator, or OSA) can introduce security vulnerabilities and possible threats to Universal Server. Universal Server is carefully designed to give the DBSSO, AAO, and Universal Server administrator 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.

Universal Server Administrator

The Universal Server administrator controls and monitors the database. The countermeasure to a threat from the Universal Server administrator is independent scrutiny of the DBMS audit trail because auditing all Universal Server administrator actions is enabled by the DBSSO or the AAO.

Database System Security Officer (DBSSO)

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 (OSA)

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 (AAO)

The AAO reviews the DBMS audit trail. The countermeasure to this threat is to ensure that an AAO is authorized to view information that may 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.

Shared-Memory Connection Threats

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 overindexes data arrays.

The OSA ensures that the shared-memory connection method is not specified in the configuration file **$INFORMIXDIR/etc/sqlhosts** 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.

Introduced Malicious Software Threats

A regular user might inadvertently execute malicious software, such as a Trojan Horse. This software might, for example:

- 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, the following steps are recommended:

- 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 that he or she has granted remote access to privileged data when he or she 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 his or her 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 Universal Server administrator when a user account is removed from the operating system. The Universal Server administrator 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 untrusted software is executed by Universal Server administrators. This untrusted software could use the privileges of the Universal Server administrator 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 database administrators do not execute software of unknown or untrusted origin. Informix further recommends that the operating-system access controls protect all software that is executed by database administrators against unauthorized modification.

Distributed-Databases Configuration Threats

When you set up distributed databases, you configure two or more software installations. These configurations could be incompatible.

Thus, a user might be able to gain access to data on a remote system with an incompatible configuration when that data would not be accessible if the same user tried to access it directly on the remote system. In the worst case, two systems could be connected by the software, each with an account with the same user name but owned by a different user. Each user is granted the privileges of the other user when he or she accesses a database that resides on the host computer of the other user.

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

Audit Administration

This chapter explains how to set up and administer auditing on your Universal Server, assuming that Universal Server is installed and functions properly. It 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 Universal Server administrator
- The Database System Security Officer (DBSSO)
- The Audit Analysis Officer (AAO)

This section also touches on the roles and responsibilities of database administrators, operating-system administrators, and system users; tells how to set up role separation; and provides guidelines for how to assign roles.

The Universal Server Administrator

The Universal Server administrator configures, maintains, and tunes Universal Server. The Universal Server administrator becomes involved with the security of a database server during installation. The overall role of the Universal Server administrator is defined in the *INFORMIX-Universal Server Administrator's Guide*.

This role should be performed by someone who has the appropriate permissions to view all the data on a database server. It is supported by a designated account and software designed to support Universal Server administrator tasks.

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

*Tip: A Universal Server administrator is any user who belongs to the group **informix**, with or without role separation.*

The 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, as described in Chapter 4, "Utility Syntax."

The DBSSO role is supported by a designated account and software. To use the audit tools, the person who fills the DBSSO role must log in to the designated account and meet access-control requirements. Once the DBSSO meets the access-control requirements and uses the administrative software, his or her actions can be audited.

*Tip: A DBSSO is any user who belongs to the group that owns **$INFORMIXDIR/ dbssodir**.*

*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, turning off auditing for a specific user. To reduce this risk, all actions taken through **onaudit** should be audited.*

The Audit Analysis Officer

The AAO configures auditing and reads and analyzes the audit trail.

The AAO configures auditing and can specify whether and how auditing is enabled, how the system responds to error conditions, and who is responsible for managing the audit trail. For Universal Server-managed auditing, the AAO also determines the directory for the audit trail and the maximum size of the audit file. For operating-system-managed auditing, the AAO should coordinate with the operating-system administrator how to read the data from the operating-system audit trail.

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

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

The installation script for Universal Server creates an **$INFORMIXDIR/aaodir** directory, which contains files that the AAO uses. These files include the **adtcfg** audit configuration file as well as the **adtcfg.std** file, which contain examples of valid definitions for audit configuration parameters.

The AAO needs appropriate permissions 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 Universal Server administrator.

Tip: An AAO is any user who belongs to the group that owns $INFORMIXDIR/ aaodir.

Other Administrative Roles and Users

A number of other, more minor, roles might be involved in Universal Server secure auditing. These roles are discussed briefly in this section.

The Database Administrator

The database administrator (DBA) manages access control for a specific database. The 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 *Informix Guide to SQL: Syntax* and *Informix Guide to SQL: Tutorial*. For information on this and other Universal Server roles and users, see the *INFORMIX-Universal Server Administrator's Guide*.

The Operating System Administrator

The OSA carries out responsibilities and tasks that Universal Server requires from the operating system. The OSA enables role separation, grants and revokes access to and from Universal Server if role separation is enforced, and adds new AAO, DBSSO and Universal Server administrator 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 Universal Server, and no special Universal 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 Universal Server administrator, DBSSO, AAO, and the account called **informix**, potentially can use Universal Server. All users with accounts who want to use Universal Server must explicitly be granted access to Universal Server if role separation is configured to enforce access control on Universal Server users. That access can be revoked 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 Universal Server recognizes as having additional privileges and/or responsibilities. These privileged users include the Universal Server administrator, 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 Universal 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 than the person who monitors the audit trail (AAO), and both should be different than the person who is responsible for the operations of the database server (Universal Server administrator).

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 Universal Server administrator 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 Universal Server administrator and DBSSO accounts as possible.

  The Universal Server administrator 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 Universal Server administrator and DBSSO roles separate.

  You might not have the resources or see the need to have the Universal Server administrator and DBSSO roles performed by different users, nor is it strictly required by Universal Server. However, when you keep the Universal Server administrator and DBSSO roles separate, you constrain them to perform only those tasks specified by their duties and limit the risk of compromising security.

- Keep the AAO role separate from the Universal Server administrator and DBSSO roles.

  The AAO determines whether to audit all Universal Server administrator or DBSSO actions in the system. It is essential that someone with a role different from that of the Universal Server administrator and DBSSO be in charge of auditing configuration, so that all users including the Universal Server administrator 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 Universal Server access-control mechanisms.

Configuring and Enforcing Role Separation

Role separation is configured during Universal Server installation. 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.

The Universal Server administrator, or the person who installs Universal Server, enforces role separation and decides which users will be the DBSSO and AAO. To find the group of the Universal Server administrator, DBSSO, and AAO, you must look at the appropriate directory of **$INFORMIXDIR**.

- Users who can perform the Universal Server administrator role are members of the group that owns the directory **$INFORMIXDIR/etc**.

- Users who can perform the DBSSO role are members of the group that own the **$INFORMIXDIR/dbssodir** directory.

- Users who can perform the AAO role are members of the group that own the **$INFORMIXDIR/aaodir** directory.

The `ls -lg $INFORMIXDIR` command produces the sample output shown in Figure 2-1.

```
total 14
drwxrwxr--  2 aao1     ix_aao      512 Nov 21 09:56 aaodir/
drwxr-xr-x  2 informix informix   1536 Nov 30 18:35 bin/
drwxrwxr-x  2 dbsso1   ix_dbsso    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
Sample 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 Universal Server administrator 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 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.

For information on environment variables, see the *Informix Guide to SQL: Reference*. For information on configuring role separation, see the *UNIX Products Installation Guide*.

Setting Up Auditing

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

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

Setting Up the Default and Compulsory Masks

Before you set up default and compulsory masks, the DBSSO needs to understand how the various masks work and what are the implications of different auditing instructions. See Chapter 1, "What Is Auditing?" for details.

Also, be sure you understand which auditing events to place in which masks.

Use the **onaudit** utility to add audit events to audit masks. The audit events and their mnemonics are listed in Appendix A, "Audit Events." The complete syntax for **onaudit** is shown in Chapter 4.

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. See Chapter 4 for specifics.

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

Specifying a Directory for Audit Files

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

If you specify that Universal Server store audit records, as described in Chapter 1, audit files are stored in a file-system directory. You can specify the directory with the **onaudit** utility. The following command specifies /**work/audit** as the directory in which audit files are to be written:

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

You can change the audit directory at any time. See Chapter 4 for more information. You also can set up the type of auditing and specify the directory with the **ADTCFG** file, shown in Appendix B.

Setting the Error Mode

As described in Chapter 1, Universal Server has four actions it can perform if it experiences an error when writing to an audit file—a *continue* error mode, and three levels of severity of *halt* error mode. Be sure 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. See Chapter 4 for the **onaudit** syntax and Appendix B for the ADTERR configuration parameter.

The following **onaudit** command sets the error mode to *continue*. Universal 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*, where Universal 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.

The AAO handles all audit-record management. If operating-system auditing is used, before you can configure auditing, operating-system auditing must be configured to accept Universal Server audit data.

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

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

To ensure that the appropriate Universal Server activities are monitored, review the audit records that are stored in the operating-system audit trail or Universal Server audit files. Universal Server must be configured to monitor these events; you can reconfigure auditing as usage changes and potential security threats are identified. For the **onaudit** syntax, see Chapter 4, "Utility Syntax." For information on the ADTMODE configuration parameter, see Appendix B, "The ADTCFG File."

Important: Although Universal Server audit-record generation might have a negative effect on Universal Server performance and resources, it nevertheless is advisable to perform more than the minimal Universal Server audit. This improves the likelihood that you will detect security violations and any attempts to circumvent security mechanisms.

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

You should 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 and/or events audited
- Processor configuration
- System load (number of processes and users)
- Disk space
- Work load (types of processes performed)

For complete information on Universal Server performance considerations, see the *INFORMIX-Universal Server Performance Guide*.

Activating Auditing

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

The following **onaudit** command turns on auditing; see Chapter 4 for the complete syntax:

```
onaudit -l 1
```

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

The AAO can configure Universal Server to turn auditing on when shared memory is initialized when the ADTMODE parameter is set to a number from 1 through 8 in the **ADTCFG** file. For details on ADTMODE parameter values, see "Changing the Auditing Configuration" on page 4-14 and Appendix B.

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

To turn off auditing once it starts, see "Turning Off Auditing" on page 2-22.

> *Important: Informix recommends 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 Universal Server.*

Maintaining Audit Masks

You might want to change the auditing instructions as your auditing needs change. This chapter 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, performed by the DBSSO, apply whether the audit records are administrated by Universal Server or by your operating system.

Creating Audit Masks

You can create masks that more closely match the types of activities performed by individual users than do default and compulsory 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 (_). Template and user masks are described in Chapter 1.

You specify events in the mask when you create it, using the audit events from the alphabetical listing in the table "Audit-Event Mnemonics" on page A-1. 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; the syntax is shown in Chapter 4. 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 a base, which makes it easier to create individual user masks.

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 also can use existing user, **_default**, **_require**, or **_exclude** masks as base masks.

Tip: If you use a template or user mask as a base for another mask, the new mask inherits the events in the base mask. It 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
```

See Chapter 4 for the syntax for creating a user mask and another example.

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 a sample input file. The syntax for the input file is explained in Chapter 4.

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

Figure 2-2
A Sample Input File

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

- The first line shows the auditing instructions for the template **_secure1**. The version that is added has the same audit events as the old version, with the addition of failed attempts to start a new session (FSTSN). All further instances of **_secure1** in this file refer to the new version of the mask.

- In the second line, the instructions specify auditing for user **kickt** in the new template **_secure1**.

- The third 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 fourth line, the user **pat** is audited with the instructions that are specified in the template **_secure2**, with the addition of all attempts at Alter Table (ALTB), and excluding all attempts at Create Table (CRTB), Create Index (CRIX), and Start New Session (STSN).

■ No template is specified for the target mask **jaym** in the fifth line, and no events are indicated; the mask is empty. (This prevents the **_default** mask from being applied to **jaym**.)

■ In the sixth 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.*

Informix provides a sample audit mask input file, **adtmasks.std**, in the **$INFORMIXDIR/aaodir** directory. The **adtmasks.std** file is intended only to serve as a guide to the DBSSO for how to set up an audit masks.

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 1-21). 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 contained in each mask. 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 they need to be modified. 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, "Utility Syntax."

> *Tip: If a base mask is used 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.)

Modifying a Mask from the Command Line

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 +SALTB,SSTB
```

When you supply a base mask with the -**r** option, it replaces 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. See Chapter 4 for the syntax and another example of how to modify a mask.

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
```

See Chapter 4 for the syntax of the **onaudit** utility.

Maintaining the Audit Configuration

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

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

Displaying the Audit Configuration

You can display the current audit configuration with the **-c** option of the **onaudit** utility. See Chapter 4 for the syntax of the **onaudit** utility.

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

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright (c) Informix Software, Inc., 1997

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

Figure 2-3
Sample Output for
onaudit -c
Command

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

- ADTMODE is set to 1, which indicates that Universal 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 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 System-Monitoring Interface (SMI) **sysadtinfo** table in the **sysmaster** database. See the *INFORMIX-Universal Server Administrator's Guide* for details.

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 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, shown in the following example, saves the current audit log to another file whenever you run it:

    ```
    onaudit -n
    ```

- Use **onaudit** **-p** to change the directory in which Universal Server writes audit files. The following example specifies **/work/audit** as the directory where the audit files are to be kept:

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

Also, a new audit file starts every time you start Informix-managed auditing.

You can use more than one flag at a time in an **onaudit** command. See Chapter 4 for the **onaudit** utility syntax used to start a new audit file, change the audit-file size, or change the pathname of the audit directory.

Changing the Audit Mode

Use the **onaudit** utility to change between operating-system-managed auditing and Universal Server-managed auditing and to change the mandatory auditing of the Universal Server administrator and/or DBSSO.

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 Universal Server administrator and DBSSO, enter the following command:

```
onaudit -l 8
```

See Chapter 4 for the syntax of the **onaudit** utility.

Changing the Audit Error Mode

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

```
onaudit -e 1
```

See Chapter 4 for the syntax of the **onaudit** utility.

Turning Off Auditing

You must use the **onaudit** utility to turn off auditing. The following example shows the command that turns auditing off:

```
onaudit -l 0
```

See Chapter 4 for the syntax of the **onaudit** utility.

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 permission to execute the utility.*
- *The user incorrectly specifies the command syntax of the utility.*
- *The utility cannot connect to shared memory.*

Audit Analysis

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

- The format of audit records produced by INFORMIX-Universal Server
- 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 Universal Server or your operating system to store and maintain the audit trail.

An overview of the audit analysis process is found in Chapter 1, "What Is Auditing?"

The Audit-Record Format

The format for Universal Server audit records has the following parts:

- The first part is an operating-system audit header, if operating-system auditing is used. It contains information supplied by the operating system, which is different for every operating system.
- The second part of the audit record is generated by Universal Server.

Universal Server audit records have the format shown in Figure 3-1 on page 3-4.

ONLN	date and time	host-name	pid	database server name	user name	errno	event mnemonic	additional fields
ONLN	1997-02-28 15:43:00.000	turk	4549	khan	jazt	0	CRDB	dbsch
ONLN	1997-02-28 15:43:18.000	turk	4549	khan	jazt	0	ACTB	dbsch:jazt:v1:103
ONLN	1997-02-28 15:43:19.000	turk	4549	khan	jazt	0	CLDB	dbsh
ONLN	1997-02-28 15:43:21.000	turk	4549	khan	jazt	0	ALFR	local:109:-:-:4:4: db1,db2,db3, rootdbs
ONLN	1997-02-28 15:43:28.000	turk	4549	khan	jazt	0	ALFR	local:109:aa5x:-:32:4: db1,db2
ONLN	1997-02-28 15:43:29.000	turk	4549	khan	jazt	0	STDS	2:-
ONLN.	1997-02-28 15:43:29.000	turk	4549	khan	jazt	0	STPR	100
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

ONLN	is a fixed field used to identify Universal Server events.
date and time	indicates when the audit event was recorded.
hostname	is the host computer of the client that executes the audit event.
pid	is the process ID of the client program that causes Universal Server to execute the audit event.
database server name	is the name of the database server on which the audit event is executed.
user name	is the login name of the user who requests the event.
errno	is the event result that contains the error number returned by the event, indicating success (0) or failure.
event mnemonic	indicates the Universal Server audit event that was executed by the user; for example, ALFR (Alter Fragment).
additional fields	indicates any additional fields used to identify databases, tables, and so on. (In a sense, the Universal Server audit record is an additional field for the operating-system audit record.) Additional fields are listed in Appendix A, "Audit Events."

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 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** to analyze it. You can also choose to put the data in a database and analyze it with SQL, as described in the next section.

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

The following example extracts audit records for the user **pat** from the Informix-managed audit file named **laurel.12**, and sends it 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, "Utility Syntax."

Audit Analysis With SQL

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

Preparing for SQL Audit Analysis

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 into an output file. The following example extracts audit records for the user **pat** from the Universal Server-managed audit file **laurel.12** and directs the records into the **records_pat** output file:

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

Create 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 also can create an ANSI-compliant database. Although an ANSI-compliant database has the additional overhead of logging, its treatment of table permissions makes it attractive in a secure environment. For more information about permissions, refer to "Privileges to Protect Audit Data" on page 3-8.

The following SQL statement creates an ANSI-compliant database:

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

Create a Table

Create a table to hold audit data with the CREATE TABLE statement. The order and data types of the columns are important; follow the example in Figure 3-2. The sample schema reflects the format of the **dbload** data file created by **onshowaudit**.

The sample CREATE TABLE statement shown in Figure 3-2 creates an audit table with the name **frag_logs**. It has **objname** and **partno** fields that are used to store fragmentation information in the event records.

```
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 3-2
Sample CREATE TABLE Statement for Audit Table

The table created in Figure 3-2 does not have any indexes on it. To improve audit-analysis performance, place indexes on columns within the table, depending on the type of analysis you perform. See the *INFORMIX-Universal Server Performance Guide* for guidance on indexing columns.

In the audit table created in Figure 3-2 on page 3-7, the **objname** (object name) consists of the **idxname** (index name) and **tabname** (table name), or another similar identifier. The **partno** consists of the **partno** (partition number or frag_id, which is always the physical partition of the record) and **oldpartno** (old partition number). Additionally, the **row_num** represents the **row_num** (row number, which is always the physical rowid for the record being manipulated), **rowid** (row identifier), and **oldrowid** (old row identifier).

Each row for the events Read Row (RDRW), Insert Row (INRW), Update Current Row (UPRW), and Delete Row (DLRW) is identified by the physical fragment and row identifier. Every audit record for these events contains the partition number and the physical row number. Table fragmentation or the existence of a unique rowid in the fragmented tables does not affect the entry in the **partno** and **row_num** fields in the audit records.

Privileges to Protect Audit Data
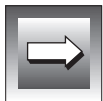
Important: Tables created in databases that are not ANSI-compliant have privileges that allow all users access. Although the default database permissions prevent access to the tables, proper security practice protects the audit-analysis table in a database that is not ANSI-compliant by revoking access 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. See the *Informix Guide to SQL: Syntax* for information on SQL statements.

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

You also can 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. See the *Informix Guide to SQL: Reference* for details.

Creating a Command File for dbload

To load the audit information into the table 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 created by **onshowaudit**. Command files and their use with **dbload** are described in the *Informix Migration Guide*.

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

delimiter | |
no. 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**. It includes the **records_pat** data file created in "Creating a Data File for dbload" on page 3-6 and the **frag_logs** table created in "Create a Table" on page 3-7:

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

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

Loading Audit Data into a Universal Server Database

Once 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 "Create a Database" on page 3-6:

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

Once the data has been loaded, begin your audit analysis with SQL.

SQL Audit-Analysis Considerations

When you plan audit analysis with Universal 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 Universal Server.

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

Whether you perform analysis with DB-Access or build a customized application, remember the advice given for audit review in Chapter 1. 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 Universal 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 Chapter 1.

Utility Syntax

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

- The **onaudit** utility performs the following operations:
    - Displays audit masks
    - Creates audit masks
    - Modifies audit masks
    - Deletes audit masks
    - Starts a new audit trail
    - Shows the audit configuration
    - Changes global auditing activities
    - Enables and disables auditing
    - Sets the error mode
    - Sets the directory in which audit files reside
    - Specifies the maximum size for an audit file
    - Determines whether Universal Server or the operating system manages the audit trail
    - Establishes mandatory auditing for various administrative roles
- The **onshowaudit** utility performs the following operations:
    - Extracts audit information from audit trail
    - Prepares extracted audit data for **dbload** to use

The onaudit Utility

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

onaudit ──

| Showing Audit Masks |
| p. 4-5 |

| Creating or Adding an Audit Mask |
| p. 4-6 |

| Modifying an Audit Mask |
| p. 4-10 |

| Deleting Audit Masks |
| p. 4-11 |

| Starting a New Audit File |
| p. 4-12 |

| Showing the Auditing Configuration |
| p. 4-13 |

| Changing the Auditing Configuration |
| p. 4-14 |
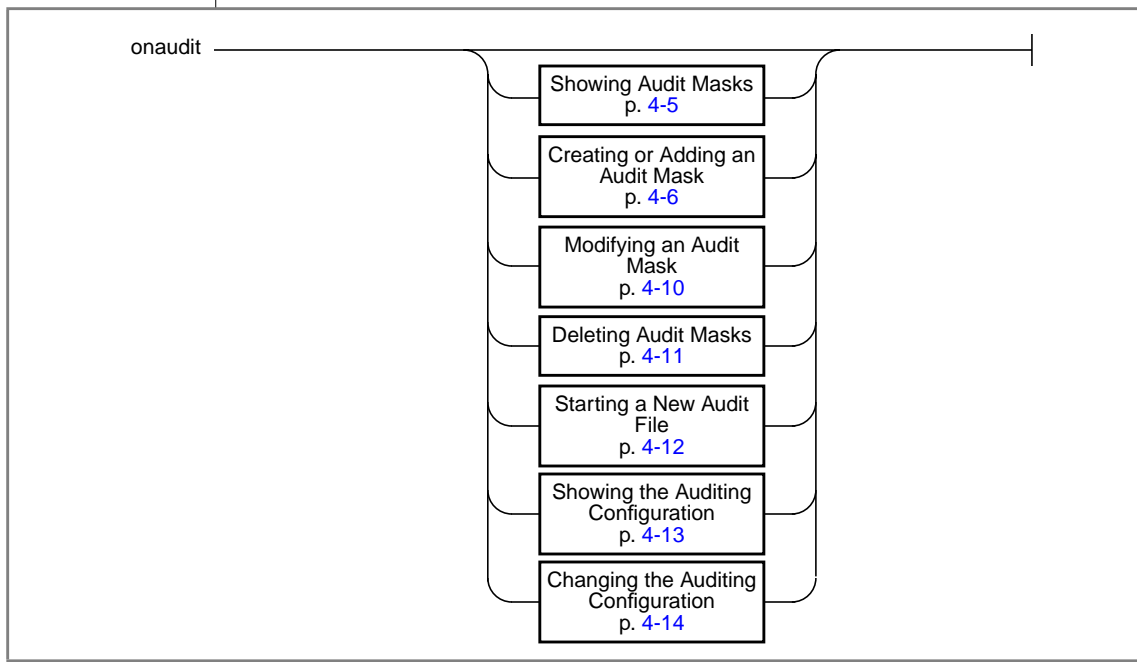
If your system has role separation, only the DBSSO or AAO can run the **onaudit** utility. The DBSSO can perform only **onaudit** functions that involve audit masks, and the AAO can perform only **onaudit** functions that involve audit configuration parameters. Without role separation, the user **informix** and **root** can perform all these tasks.

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

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

Showing Audit Masks

```
┌─────────────────────┐
│ Showing Audit Masks │
└─────────────────────┘

    ──────────────── -o ────────────────────────────────────►

                              ┌─── -u mask ───┐
                              └───── -y ───────┘
```
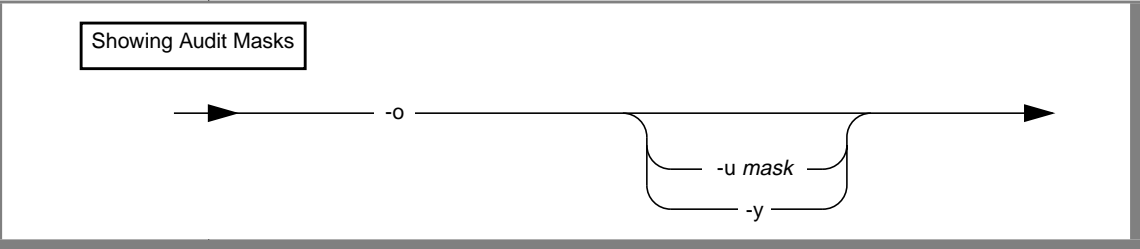
Element	Purpose	Key Considerations
-o	Outputs audit masks.	None.
-u *mask*	Names a specific mask to display.	**Additional Information**: *Mask* can be any existing mask.
-y	Automatically responds `yes` to the confirmation prompt.	None.

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. It is the same format as that of an input file for **onaudit**, as described in "Modifying an Audit Mask" on page 4-10.

```
maskname      basemask      audit_events
```

Because Universal Server keeps no record of the base mask that is used to create or modify a mask, a single dash (-) 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
```

Creating or Adding an Audit Mask

```
┌──────────────────────┐
│ Creating or Adding an│
│      Audit Mask      │
└──────────────────────┘
                                    ┌──────────────────────┐
                                    │   The Audit-Mask      │
              ──────────────  -a ───│   Specification       │──────────────────►
                                    │      p. 4-7           │
                                    └──────────────────────┘
                                    ┌──────────────────────┐
                                    │ The onaudit Input-File│
                     └───────  -f ──│      Format           │
                                    │      p. 4-9           │
                                    └──────────────────────┘
```
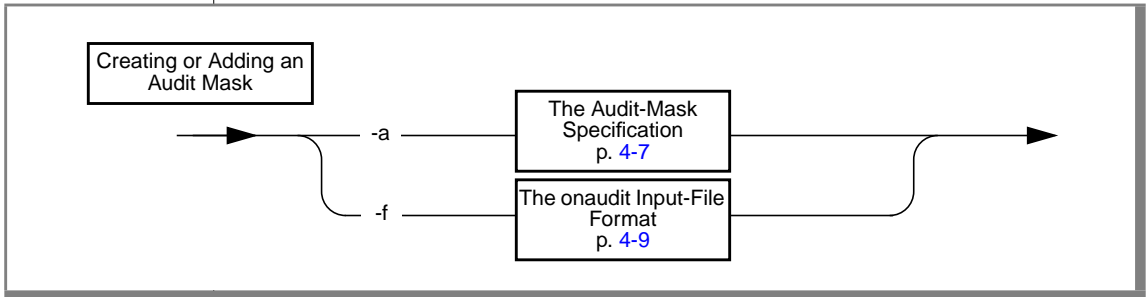
Element	Purpose	Key Considerations
-a	Adds a new audit mask.	None.
-f	Names a file that can include instructions to add any or all of the audit masks to the mask table.	**References:** The syntax for the input file is described in "The onaudit Input-File Format" on page 4-9.

The Audit-Mask Specification

Element	Purpose	Key Considerations
+	Indicates that the events that follow are to be added to the list of audit events in *targetmask*.	**Additional Information**: The + is the default and thus is optional.
−	Indicates that the events that follow are to be removed from the list of audit events in *targetmask*.	None.
-e	Indicates that the arguments that follow specify audit events to be added to or removed from *targetmask*.	**Additional Information**: Events specified as arguments to -**e** override events listed in any base mask specified with the -**r** option.

(1 of 2)

Element	Purpose	Key Considerations
-r *basemask*	Names an existing audit mask to use as a base when you create or modify a *targetmask*. The current events listed in the base mask are applied to the target mask.	**Additional Information**: If no *basemask* is specified and no events are specified with the **-e** flag, **onaudit** creates an empty target mask. If a *basemask* subsequently changes, those changes are not reflected in masks for which it has been used as a base. Target masks are created with a snapshot of the base mask only; they do not dynamically refer to that base mask.
-u *targetmask*	Names a user, template, **_default**, **_require**, or **_exclude** mask to be created or modified.	**Restrictions**: The *targetmask* must be eight characters or less.
F*event*	Specifies that only failed event attempts are to be audited.	**Additional Information**: The *event* can include the event code (mnemonic) for any event listed in the table "Audit-Event Mnemonics" on page A-1.
S*event*	Specifies that only successful event attempts are to be audited.	**Additional Information**: The *event* can include the event code (mnemonic) for any event listed in the table "Audit-Event Mnemonics" on page A-1.
event	Names an event to audit, whether the event execution succeeds or fails.	**Additional Information**: The *event* can include the event code (mnemonic) for any event listed in the table "Audit-Event Mnemonics" on page A-1.

(2 of 2)

Tip: Do not include any spaces in the events list or you might get unpredictable results.

The following example creates a new audit mask for the user **pat**. The new mask audits the events specified in the **_secureL** template mask, with the exception of Read Row (RDRW) and with the addition of 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, "What Is Auditing?"

The onaudit Input-File Format

```
The onaudit Input-File
Format

──►── targetmask ──┬── basemask ──┬─────────────────────────────►──
                   └──── - ────┘   │            ┌──── , ────┐
                                   ├──/1─┬───┬──┬── event ──┤
                                   │     └─+─┘  ├── Fevent ─┤
                                   └──/1── - ───┴── Sevent ─┘
```

Element	Purpose	Key Considerations
+	Indicates that the 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.
F*event*	Specifies that only failed event attempts are to be audited.	**Additional Information**: The *event* can include the event code (mnemonic) for any event listed in the table "Audit-Event Mnemonics" on page A-1.
S*event*	Specifies that only successful event attempts are to be audited.	**Additional Information**: The *event* can include the event code (mnemonic) for any event listed in the table "Audit-Event Mnemonics" on page A-1.
basemask	Names an existing audit mask to use as a base.	**Additional Information**: The auditing instructions of the base mask are copied to the target mask, in addition to (or except for) the audit events that follow.
event	Names an event to audit.	**Additional Information**: The *event* can include the event code (mnemonic) for any event listed in the table "Audit-Event Mnemonics" on page A-1.
targetmask	Identifies the user, template, **_default**, **_require**, or **_exclude** mask to add.	**Restrictions**: Mask names must not exceed eight characters, and template mask names must begin with an underscore (_).

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 a sample **onaudit** input file, see Chapter 2, "Audit Administration."

Modifying an Audit Mask

```
┌─────────────────────────────────┐
│  ┌──────────────────┐           │
│  │ Modifying an Audit │           │
│  │      Mask          │           │
│  └──────────────────┘           │
│                    ┌──────────────┐
│  ─────────── -m ───│ The Audit-Mask │───────►
│                    │ Specification  │
│                    │    p. 4-7      │
│                    └──────────────┘
└─────────────────────────────────┘
```
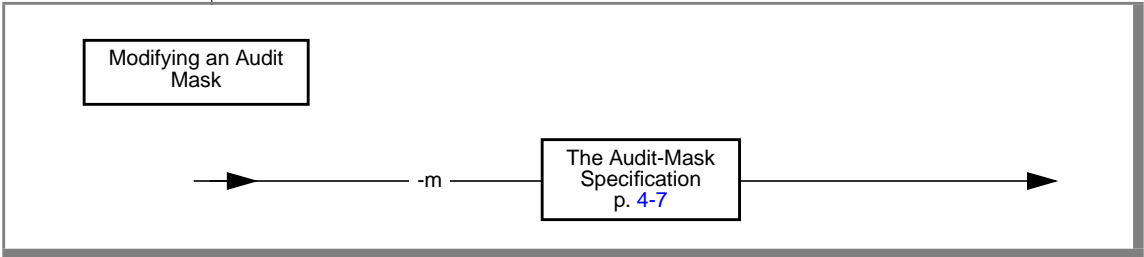
Element	Purpose	Key Considerations
-**m**	Modifies an existing audit mask.	None.

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
```

Deleting Audit Masks

| Deleting Audit Masks |

```
                ──────────────── -d ────────────────────────▶
                                        ┌─── -u mask ───┐
                                        └───── -y ──────┘
```
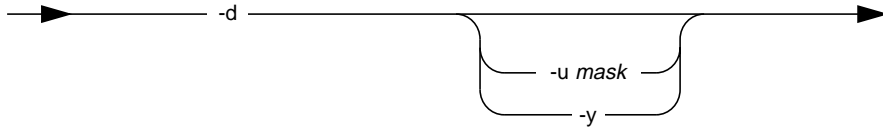
Element	Purpose	Key Considerations
-**d**	Deletes audit masks.	None.
-**u** *mask*	Names a specific mask to delete.	**Additional Information**: *Mask* can be any existing mask.
-**y**	Automatically responds yes to the confirmation prompt.	None.

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

- 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

Starting a New Audit
File

────────► ── -n ──►

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

Storing Audit Files on Universal Server with the -n Option

If you store the audit records on Universal Server, the -**n** option to the
onaudit utility closes the current audit file and opens a new one. The audit
file is named according to the naming convention *servername.integer*. The
servername 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**.

Storing Audit Files on the Operating System with the -n Option

If you store the audit records on the operating system, the -**n** option to the
onaudit utility closes the current audit log file and opens a new one. See your
operating-system documentation for the naming convention.

Showing the Auditing Configuration

```
Showing the Auditing
Configuration
```
`-c`

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

The **-c** option directs the **onaudit** utility to display the current state of auditing. Figure 4-1 shows sample audit-configuration output.

Figure 4-1
Sample
Audit-Configuration
Output

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright (c) Informix Software, Inc., 1997

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

You can change the auditing configuration, as described in the next section.

Changing the Auditing Configuration

Element	Purpose	Key Considerations
-e *error mode*	Specifies the error-handling method for auditing when Universal Server cannot write a record to the audit file.	**Restrictions**: The *error mode* can have one of the following values: 0, 1, 3.
		Additional Information: This option pertains to the value set for the ADTERR parameter in the **ADTCFG** file and 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 4-15.
-l *audit mode*	Specifies the auditing mode.	**Restrictions:** The *audit mode* can have one of the following values: 0, 1, 2, 3, 4, 5, 6, 7, 8.
		Additional Information: This option pertains to the value set for the ADTMODE parameter in the **ADTCFG** file. For details of the valid *audit mode* values, see "Using the -l Option" on page 4-16.
-p *auditdir*	Names the directory where the audit files are to be created.	**Restrictions**: You can change the *auditdir* only when Informix-managed auditing is on.
		Additional Information: This option pertains to the value set for the ADTPATH parameter in the **ADTCFG** file. The change occurs with the next write attempt. Universal Server starts a new audit file in the new directory, beginning with the first available number that is equal to or greater than 0.

(1 of 2)

Element	Purpose	Key Considerations
-**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 change the *maxsize* only when Informix-managed auditing is on.
		Additional Information: This option pertains to the value set for the ADTSIZE parameter in the **ADTCFG** file. When an audit file reaches or exceeds *maxsize*, the current file is closed and a new audit file is started.

(2 of 2)

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

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**.

The value 0 is also known as *continue mode*. It indicates that Universal Server is to continue processing the thread and note the error in the message log. Errors for subsequent attempts to write to the audit file are also sent to the message log.

The -**e** option has the following *halt mode*s:

- 1 indicates that Universal Server is to suspend processing a thread when it cannot write a record to the current audit file and is to continue the write attempt until it succeeds.

- 3 indicates that Universal Server is to shut down.

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 auditing *off*. Universal Server stops auditing for all existing sessions, and new sessions are not audited.

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

- 1 turns on Universal Server-managed auditing for all sessions but does not automatically audit DBSSO and Universal Server administrator actions.

- 2 turns on operating-system-managed auditing but does not automatically audit DBSSO and Universal Server administrator actions.

- 3 turns on Universal Server-managed auditing and automatically audits DBSSO actions.

- 4 turns on operating-system-managed auditing and automatically audits DBSSO actions.

- 5 turns on Universal Server-managed auditing and automatically audits Universal Server administrator actions.

- 6 turns on operating-system-managed auditing and automatically audits Universal Server administrator actions.

- 7 turns on Universal Server-managed auditing and automatically audits DBSSO and Universal Server administrator actions.

- 8 turns on operating-system-managed auditing and automatically audits DBSSO and Universal Server administrator actions.

The onshowaudit Utility

The **onshowaudit** utility lets you extract information from an audit trail, and you can specify that it do so for particular users and/or database servers. 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 for use with the **dbload** utility.

The **onshowaudit** utility extracts data from an audit trail, but it does not process the records or delete them from the audit trail.

If your system has role separation, only the AAO can run the **onshowaudit** utility. Without role separation, only user **informix** and **root** can run **onshowaudit**.

onshowaudit ─┬─────┬─┬─────────────┬─┬───────────────┬─┬───────────────────┬─┬──────┤
             ├ -I ─┤ └─ -f *path* ─┘ └─ -u *username* ─┘ └─ -s *servername* ─┘ └─ -l ─┘
             └ -O ─┘

*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 the file as input for **dbload**. The **dbload** utility cannot process the generated header lines.*

How you specify a path (if any) determines which part of the audit trail the **onshowaudit** utility uses.

Element	Purpose	Key Considerations
-I	Uses the Informix audit trail.	None.
-O	Uses the operating-system audit trail.	None.
-f *path*	names a specific audit trail to examine, only for Informix-managed auditing.	**Additional Information**: 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.
-l	Directs **onshowaudit** to take the extracted information and reformat it for **dbload** to use.	**References**: See Chapter 3, "Audit Analysis," for information on the file format. See the *Informix Migration Guide* for information on the **dbload** utility.
-s *servername*	Names the specific database server about which to extract audit information.	None.
-u *username*	Specifies the login name of a user about which to extract audit information.	None.

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

Informix does not audit the execution of the **onshowaudit** utility. When you use operating-system-managed auditing, **onshowaudit** calls operating-system utilities to extract from the operating-system audit trail audit records that are generated by the Informix DBMS.

Important: Informix recommends that the OSA always enable auditing for utilities that extract audit events from the operating-system audit trail.

Audit Events

This appendix contains the following two tables:

- Auditable events, listed alphabetically by event mnemonic (below)

- Audit-event records and their fields (see)

Important: Universal Server secure-auditing facility audits only the events that are listed in this appendix. You might encounter additional SQL statements that are not audited.

Audit-Event Mnemonics

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

Mnemonic	Event Name
ACTB	Access Table
ADCK	Add Chunk
ADLG	Add Transaction Log
ALFR	Alter Fragment
ALIX	Alter Index
ALME	Alter Access Method
ALOP	Alter Optical Cluster

(1 of 6)

Mnemonic	Event Name
ALTB	Alter Table
BGTX	Begin Transaction
CLDB	Close Database
CMTX	Commit Transaction
CRAM	Create Audit Mask
CRBS	Create Blobspace
CRBT	Create Opaque Type
CRCT	Create Cast
CRDB	Create Database
CRDS	Create Dbspace
CRDT	Create Distinct Type
CRIX	Create Index
CRME	Create Access Method
CRNRT	Create Named Row Type
CROC	Create Operator Class
CROP	Create Optical Cluster
CRRL	Create Role
CRSN	Create Synonym
CRSP	Create Stored Procedure
CRTB	Create Table
CRTR	Create Trigger
CRVW	Create View
DLRW	Delete Row
DNCK	Bring Chunk Off-line

(2 of 6)

Mnemonic	Event Name
DNDM	Disable Disk Mirroring
DRAM	Delete Audit Mask
DRBS	Drop Blobspace
DRCK	Drop Chunk
DRCT	Drop Cast
DRDB	Drop Database
DRDS	Drop Dbspace
DRIX	Drop Index
DRLG	Drop Transaction Log
DRME	Drop Access Method
DRNRT	Drop Named Row Type
DROC	Drop Operator Class
DROP	Drop Optical Cluster
DRRL	Drop Role
DRSN	Drop Synonym
DRSP	Drop Stored Procedure
DRTB	Drop Table
DRTR	Drop Trigger
DRTY	Drop Type
DRVW	Drop View
EXSP	Execute Stored Procedure
GRDB	Grant Database Access
GRFR	Grant Fragment Access
GRRL	Grant Role

(3 of 6)

Mnemonic	Event Name
GRTB	Grant Table Access
INRW	Insert Row
LGDB	Change Database Log Mode
LKTB	Lock Table
LSAM	List Audit Masks
LSDB	List Databases
MDLG	Modify Transaction Logging
ONAU	**onaudit**
ONCH	**oncheck**
ONIN	**oninit**
ONLG	**onlog**
ONLO	**onload**
ONMN	**onmonitor**
ONMO	**onmode**
ONPA	**onparams**
ONSP	**onspaces**
ONST	**onstat**
ONUL	**onunload**
OPDB	Open Database
RDRW	Read Row
RLOP	Release Optical Cluster
RLTX	Rollback Transaction
RMCK	Clear Mirrored Chunks
RNDB	Rename Database

(4 of 6)

Mnemonic	Event Name
RNTC	Rename Table/Column
RSOP	Reserve Optical Cluster
RVDB	Revoke Database Access
RVFR	Revoke Fragment Access
RVRL	Revoke Role
RVTB	Revoke Table Access
SCSP	System Command, Stored Procedure
STCN	Set Constraint
STDF	Set Debug File
STDP	Set Database Password
STDS	Set Dataskip
STEX	Set Explain
STIL	Set Isolation Level
STLM	Set Lock Mode
STOM	Set Object Mode
STOP	Stop Statement
STPR	Set Pdqpriority
STRL	Set Role
STRT	Start Statement
STSA	Set Session Authorization
STSN	Start New Session
STTX	Set Transaction Mode
TMOP	Time Optical Cluster
ULTB	Unlock Table

(5 of 6)

Mnemonic	Event Name
UPAM	Update Audit Mask
UPCK	Bring Chunk On-line
UPDM	Enable Disk Mirroring
UPRW	Update Current Row
USSP	Update Statistics, Stored Procedure
USTB	Update Statistics, Table

(6 of 6)

Audit-Event Fields

The following table shows the audit-event information 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 Universal Server utilities use to identify audit events. The list is in alphabetical order, by acronym.
- 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, Universal Server creates several audit records to carry the complete information.mumu*

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Access Table	ACTB	dbname	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[19]			owner	frag flags[16]	dbspaces
Index, Alter	ALIX	dbname	tabid					owner[15]	cluster flag[10, 15]	index name[15]
Access Method, Alter	ALME	dbname	access method ID	access method name				access method owner		
Optical Cluster, Alter	ALOP	dbname			cluster size			owner		cluster name
Table, Alter	ALTB	dbname	old tabid		new tabid[15]	frag_id				new part-nolist[15]
Transaction, Begin	BGTX									
Database, Close	CLDB	dbname								
Transaction, Commit	CMTX									
Audit Mask, Create	CRAM							user id		
Blobspace, Create	CRBS	blobspace name						owner	mirror status[1]	media
Opaque Type, Create	CRBT	dbname		opaque type name				opaque type owner		
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	dbname								dbspace name

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Dbspace, Create	CRDS	dbspace name							mirror status[1]	
Distinct Type, Create	CRDT	dbname		distinct type name				distinct type owner		
Index, Create	CRIX	dbname	tabid	idxname		frag_id		owner	frag flags[16]	dbspacelist
Access Method, Create	CRME	dbname	access meth-od ID	access method name				access method owner		
Named Row Type, Create	CRNRT	dbname	xid of row type	named row type name				named row type owner		
Operator Class, Create	CROC	dbname	opera-tor class ID	operator class name				owner		
Optical Cluster, Create	CROP	dbname	tabid		cluster size			owner		cluster name
Create Role	CRRL	dbname		rolename						
Synonym, Create	CRSN	dbname	syn. tabid		base tabid			owner	syn. type[8]	synonym name
Stored Procedure, Create	CRSP	dbname	proc. id					owner		procedure name
Table, Create	CRTB	dbname	tabid	tabname		frag_id		owner	frag flags[16]	dbspacelist
Trigger, Create	CRTR	dbname	tabid				trigger id[15]	owner[15]		trigger name[15]
View, Create	CRVW	dbname	view tabid					owner		view name
Row, Delete	DLRW	dbname	tabid		partno	frag_id	row-num[15]			

(2 of 8)

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Chunk, Bring Off-line	DNCK				chunk number				mirror status[1]	
Disk Mirroring, Disable	DNDM				dbspace number					
Audit Mask, Delete	DRAM							user id		
Blobspace, Drop	DRBS	blobspace 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			
Database, Drop	DRDB	dbname								dbpassword
Dbspace, Drop	DRDS	dbspace name								
Index, Drop	DRIX	dbname	tabid					owner		index name
Transaction Log, Drop	DRLG				log number					
Access Method, Drop	DRME	dbname	access method ID	access method name				access method owner		
Named Row Type, Drop	DRNRT	dbname	xid of dropped type							
Operator Class, Drop	DROC	dbname		operator class name				owner		
Optical Cluster, Drop	DROP	dbname						owner		cluster name
Drop Role	DRRL	dbname		rolename						

(3 of 8)

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Synonym, Drop	DRSN	dbname	syn. tabid							
Stored Procedure, Drop	DRSP	dbname	proc. id							
Table, Drop	DRTB	dbname			tabid				drop-flags[22]	partnolist
Trigger, Drop	DRTR	dbname					trigger id			
Type, Drop	DRTY	dbname		type name				type owner		
View, Drop	DRVW	dbname	view tabid						drop-flags[22]	
Stored Procedure, Execute	EXSP	dbname	proc. id							
Grant Database Access	GRDB	dbname			privilege[6]					grantees[5]
Grant Fragment Access	GRFR	dbname	tabid	fragment	privilege[6, 15]			grantor		grantee[6, 15]
Grant Role	GRRL	dbname		rolename				grantor		grantees[5]
Grant Table Access	GRTB	dbname	tabid		privilege[6, 15]			grantor		grantee[6, 15], update columns, select columns[5, 15]
Row, Insert	INRW	dbname	tabid		partno	frag_id	rowid			
Database Log Mode, Change	LGDB	dbname							log status[7]	
Table, Lock	LKTB	dbname	tabid						lock mode[9]	
Audit Masks, List	LSAM									

(4 of 8)

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Databases, List	LSDB	dbname							scan mode[2]	
Modify Transaction Logging	MDLG								buffered-log flags[3]	
onaudit	ONAU									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
onspaces	ONSP									command line
onstat	ONST									command line
onunload	ONUL									command line
Database, Open	OPDB	dbname							exclusive-flag	dbpassword
Row, Read	RDRW	dbname	tabid		partno	frag_id	rowid[15]			
Optical Cluster, Release	RLOP	family name					volume number			
Transaction, Rollback	RLTX									

(5 of 8)

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Chunks, Clear Mirrored	RMCK				dbspace number					
Rename Database	RNDB	dbname		new dbname				user id		
Table/ Column, Rename	RNTC									
Optical Cluster, Reserve	RSOP	family name					volume number			
Revoke Database Access	RVDB	dbname			privilege[6]					revokees[5]
Revoke Fragment Access	RVFR	dbname	tabid	fragment	privilege[6, 15]			revoker		revokees[5, 15]
Revoke Role	RVRL	dbname		rolename				revoker		revokees[5]
Revoke Table Access	RVTB	dbname	tabid		privilege[6, 15]			revoker	drop-flags[22]	revokees[5, 15]
Stored Procedure, System Command	SCSP									command string
Constraint, Set	STCN	dbname							con-straint mode[12]	constraint names
Set Debug File	STDF	dbname								file path
Set Database Password	STDP	dbname						user id		
Set Dataskip	STDS								skip flags[17]	dbspacelist
Set Explain	STEX								explain flags[13]	
Isolation Level, Set	STIL				isolation level[4]					

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Set Lock Mode	STLM								wait flags[14]	
Set Object Mode	STOM	dbname	tabid		command mode flag[23]				object typeflag[24]	object names
Stop Statement	STOP	dbname	tabid							
Set Pdqpriority	STPR								prlevel[18]	
Set Role	STRL	dbname		rolename						
Start Statement	STRT	dbname	tabid		Vio_tid				Dia_tid	
Set Session Authorization	STSA	dbname						new user-name		
Start New Session	STSN									
Set Transaction Mode	STTX				operation[21]				mode flags[20]	
Optical Cluster, Time	TMOP								time flag[14]	
Table, Unlock	ULTB	dbname	tabid							
Audit Mask, Update	UPAM							user id		
Chunk, Bring On-line	UPCK				chunk number				mirror status[1]	
Disk Mirroring, Enable	UPDM				dbspace number					
Row, Update Current	UPRW	dbname	tabid		old partno	old rowid[15]			new rowid	new partno

(7 of 8)

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Stored Procedure, Update Statistics	USSP	dbname	proc. id							
Table, Update Statistics	USTB	dbname	tabid							

(8 of 8)

NOTES

[1] Mirror Status: **0** Not mirrored
     1 Mirrored

[2] Scan Mode: **1** Open
     2 Read
     3 Close

[3] Buffered Log Flag: **0** Buffering turned off
     1 Buffering turned on

[4] Isolation Level: **0** No transactions
     1 Dirty read
     2 Committed read
     3 Cursor stability
     5 Repeatable read

[5] 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 3-5 truncates the lists to 166 characters.

[6] 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. Table id refers to the procedure ID.)

     Database-Level Privileges:
     256 Connect
     512 DBA
     1024 Resource

[7] Log Status: **1** Logging on
     2 Buffered logging
     4 ANSI-compliant

[8] Synonym Type: **0** Private
     1 Public

[9] Lock Mode: **0** Exclusive
     1 Shared

[10] Cluster Flag: **0** Not cluster
     1 Cluster

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

12 Constraint Mode:
0 Deferred
1 Immediate

13 Explain Flag:
0 Explain turned off
1 Explain turned on

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

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

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

17 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

18 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

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

20 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

21 Operation:
0 Set Access Mode
1 Set Isolation Level

22 Dropflags:
0 Cascade
1 Restrict

23 Command Mode Flag
1 Disabled
2 Filtering without error
4 Filtering with error
8 Enabled

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

The ADTCFG File

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

ADTCFG Parameters

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

default value	The default value that appears in the **adtcfg.std** file
if not present	The value that is supplied if the parameter is missing from your **ADTCFG** file
units	The units in which the parameter is expressed
separators	The separator(s) that can be used when the parameter value has several parts. Do *not* use white space within a parameter value.
range of values	The valid values for this parameter
takes effect	The time at which a change to the value of the parameter actually affects the operation of Universal Server
utility	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 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 Universal Server configuration parameters, see the *INFORMIX-Universal Server Administrator's Guide*.

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, Universal 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 Universal Server encounters an error as it writes an audit record, Universal Server suspends processing of the thread until it successfully writes a record.

3 = *halt* error mode: shut down system

When Universal Server encounters an error as it writes an audit record, Universal Server shuts down.

takes effect	When **onaudit** is run to change the value, or after shared memory is initialized. ADTMODE must be non-zero (auditing is on).
utility	**onaudit** (`onaudit -e errormode`)

ADTERR specifies how Universal Server behaves when it encounters an error while it writes an audit records.

ADTMODE

default value	0
range of values	0 through 8

> 0 = auditing disabled
>
> 1 = Universal Server-managed auditing on; starts auditing for all sessions
>
> 2 = operating-system-managed auditing on
>
> 3 = Universal Server-managed auditing on; audits DBSSO actions
>
> 4 = operating-system-managed auditing on; audits DBSSO actions
>
> 5 = Universal Server-managed auditing on; audits Universal Server administrator actions
>
> 6 = operating-system-managed auditing on; audits Universal Server administrator actions
>
> 7 = Universal Server-managed auditing on; audits DBSSO and Universal Server administrator actions
>
> 8 = operating-system-managed auditing on; audits DBSSO and Universal Server administrator actions

takes effect	When **onaudit** is run to change the value, or after shared memory is initialized
utility	**onaudit** (`onaudit -l auditmode`)

ADTMODE controls whether Universal Server or the operating system manages auditing of user actions.

ADTPATH

default value	**/tmp**
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 where audit files are written.

Universal Server-managed auditing must be on to change the ADTPATH with **onaudit**.

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, it is saved and a new audit file is created.

This applies only to Universal Server-managed auditing.

Index

U

W

X