

Trusted Facility Manual

for Informix® Dynamic Server™

Informix Dynamic Server
Informix Dynamic Server, Developer Edition
Informix Dynamic Server, Workgroup Edition

Version 7.3
February 1998
Part No. 000-4348

Published by INFORMIX® Press

Informix Software, Inc.
4100 Bohannon Drive
Menlo Park, CA 94025-1032

Copyright © 1981-1998 by Informix Software, Inc. or its subsidiaries, provided that portions may be copyrighted by third parties, as set forth in documentation. All rights reserved.

The following are worldwide trademarks of Informix Software, Inc., or its subsidiaries, registered in the United States of America as indicated by “®,” and in numerous other countries worldwide:

Answers OnLine™; INFORMIX®; Informix®; Illustra™; C-ISAM®; DataBlade®; Dynamic Server™; Gateway™; NewEra™

All other names or marks may be registered trademarks or trademarks of their respective owners.

Documentation Team: Geeta Karmarkar, Jennifer Leland, Bonnie Vaughan

RESTRICTED RIGHTS/SPECIAL LICENSE RIGHTS

Software and documentation acquired with US Government funds are provided with rights as follows: (1) if for civilian agency use, with Restricted Rights as defined in FAR 52.227-19; (2) if for Dept. of Defense use, with rights as restricted by vendor's standard license, unless superseded by negotiated vendor license as prescribed in DFAR 227.7202. Any whole or partial reproduction of software or documentation marked with this legend must reproduce the legend.

Table of Contents

Introduction

About This Manual	3
Types of Users	3
Software Dependencies	4
Assumptions About Your Locale.	4
Demonstration Database	5
New Features	5
Documentation Conventions	6
Typographical Conventions	6
Icon Conventions	7
Command-Line Conventions	8
How to Read a Command-Line Diagram	10
Additional Documentation	11
On-Line Manuals	11
Printed Manuals	12
Error Message Files	12
Documentation Notes, Release Notes, Machine Notes	12
Related Reading	14
Compliance with Industry Standards	14
Informix Welcomes Your Comments	15

Chapter 1

What Is Auditing?

Secure-Auditing Facility	1-3
Audit Events	1-4
Audit Masks.	1-4
Auditing Process	1-7
Audit Trail	1-8
Administrative Roles	1-9

Audit Masks and Audit Instructions	1-10
User Masks	1-10
Making a Mask for User informix	1-11
Template Masks	1-11
Setting the Audit Instructions	1-12
Audit Configuration	1-17
Turning Auditing On or Off	1-17
Specifying Types of Auditing	1-18
Properties of Audit Files on UNIX Systems	1-19
Windows NT Event Log	1-21
Windows NT Message Server	1-22
Access to the Audit Trail	1-22
Error Modes for Writing to an Audit File or the Event Log	1-23
Audit Configuration and the ADTCFG File.	1-24
Access to the Audit Trail	1-25
Audit Analysis	1-27
The Importance of Audit Analysis	1-27
Preparing for Audit Analysis	1-28
Strategies for Audit Analysis.	1-30
Responding to Identified Security Problems	1-32
DBMS Security Threats	1-33
Primary Threats	1-33
Privileged Activity Threats	1-34
Shared-Memory Connection Threats on UNIX Systems	1-35
Introduced Malicious Software Threats	1-35
Remote-Access Threats.	1-36
Obsolete-User Threats	1-36
Untrusted Software Used in a Privileged Environment.	1-37
Distributed-Databases Configuration Threats	1-37

Chapter 2 **Audit Administration**

Administrative Roles and Role Separation	2-3
The Database Server Administrator	2-4
The Database System Security Officer.	2-4
The Audit Analysis Officer	2-5
Other Administrative Roles and Users	2-6
Role Separation	2-7
Setting Up Auditing	2-10
Setting Up the Default and Compulsory Masks	2-10
Specifying a UNIX Directory for the Audit Trail	2-11
Setting the Error Mode	2-11
Setting the Audit Level.	2-12
Activating Auditing.	2-13

	Maintaining Audit Masks	2-14
	Creating Audit Masks	2-14
	Displaying Audit Masks	2-17
	Modifying Audit Masks	2-18
	Deleting Audit Masks	2-19
	Maintaining the Audit Configuration	2-19
	Displaying the Audit Configuration	2-20
	Starting a New Audit File on a UNIX System	2-21
	Changing the Audit Mode on a UNIX System	2-22
	Changing the Audit Mode on a Windows NT System	2-22
	Changing the Audit Error Mode	2-23
	Turning Off Auditing	2-23
Chapter 3	Audit Analysis	
	The Audit-Record Format	3-3
	Audit Analysis Without SQL	3-5
	Audit Analysis with SQL	3-6
	Preparing for SQL Audit Analysis	3-6
	Creating a Data File for dbload	3-6
	Creating a Database and Table for Audit Data	3-7
	Creating a Command File for dbload	3-9
	Loading Audit Data into a Database	3-10
	SQL Audit-Analysis Considerations	3-10
Chapter 4	Utility Syntax	
	The onaudit Utility	4-4
	Showing Audit Masks	4-5
	Creating or Adding an Audit Mask	4-6
	Modifying an Audit Mask	4-10
	Deleting Audit Masks	4-11
	Starting a New Audit File on a UNIX System	4-12
	Showing the Auditing Configuration	4-13
	Changing the Auditing Configuration	4-14
	The onshowaudit Utility	4-17
Appendix A	Audit Events	
Appendix B	The ADTCFG File	
	Index	

Introduction

About This Manual	3
Types of Users	3
Software Dependencies	4
Assumptions About Your Locale	4
Demonstration Database	5
New Features	5
Documentation Conventions	6
Typographical Conventions	6
Icon Conventions	7
Comment Icons	7
Feature, Product, and Platform Icons	8
Command-Line Conventions	8
How to Read a Command-Line Diagram	10
Additional Documentation	11
On-Line Manuals	11
Printed Manuals	12
Error Message Files	12
Documentation Notes, Release Notes, Machine Notes	12
Related Reading	14
Compliance with Industry Standards	14
Informix Welcomes Your Comments	15

R

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

About This Manual

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

This manual is not a computer security or trusted facility administration training manual. For detailed information on those topics, see the suggested material in [“Related Reading” on page 14](#).

Types of Users

This manual is for the following users:

- System administrators
- Database administrators
- Users of Dynamic Server who are interested in secure auditing

This manual assumes that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts

- An understanding of system administration
- A familiarity with the SQL statements that pertain to the events that you want to audit

If you have limited experience with relational databases, SQL, or your operating system, refer to [Getting Started with Informix Dynamic Server](#) for a list of supplementary titles.

Software Dependencies

This manual assumes that your database server is one of the following products:

- Informix Dynamic Server, Version 7.3
- Informix Dynamic Server, Developer Edition, Version 7.3.
- Informix Dynamic Server, Workgroup Edition, Version 7.3.

Assumptions About Your Locale

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

This manual assumes that you 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 considerations related to GLS locales, see the [Informix 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. You can use SQL scripts provided with DB-Access to derive a second database, called **sales_demo**. This database illustrates a dimensional schema for data-warehousing applications. Sample command files are also included for creating and populating these databases.

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 the [Informix Guide to SQL: Reference](#).

The scripts that you use to install the demonstration databases reside in the `$INFORMIXDIR/bin` directory on UNIX platforms and the `%INFORMIXDIR%\bin` directory on Windows NT platforms. For a complete explanation of how to create and populate the **stores7** demonstration database, refer to the [DB-Access User Manual](#). For an explanation of how to create and populate the **sales_demo** database, refer to the [Informix Guide to Database Design and Implementation](#).

New Features

Most of the new features for Version 7.3 of Informix Dynamic Server fall into five major areas:

- Reliability, availability, and serviceability
- Performance
- Windows NT-specific features
- Application migration
- Manageability

Several additional features affect connectivity, replication, and the optical subsystem. For a comprehensive list of new features, see the release notes for your database server.

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.
<i>italics</i>	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.
<code>monospace</code>	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 <i>warning</i> icon identifies vital instructions, cautions, or critical information.
	The <i>important</i> icon identifies significant information about the feature or operation that is described.
	The <i>tip</i> icon identifies additional details or shortcuts for the functionality that is described.

Feature, Product, and Platform Icons

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

Icon	Description
GLS	Identifies information that relates to the Informix GLS feature.
IDS	Identifies information that is specific to Dynamic Server and its editions. However, in some cases, the identified section applies only to Informix Dynamic Server and not to Informix Dynamic Server, Workgroup and Developer Editions. Such information is clearly identified.
UNIX	Identifies information that is specific to the UNIX platform.
W/D	Identifies information that is specific to Informix Dynamic Server, Workgroup and Developer Editions.
WIN NT	Identifies information that is specific to the Windows NT environment.

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

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.




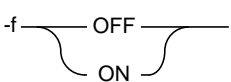
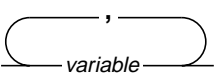
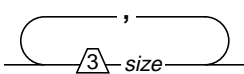
Dynamic Server supports a variety of command-line options. For example, the **onaudit** and **onshowaudit** utilities described in [Chapter 4, “Utility Syntax,”](#) require you to issue various commands.

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.
<i>variable</i>	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.
<div style="border: 1px solid black; padding: 2px; display: inline-block;">Privileges p. 5-17</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Privileges</div>	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)

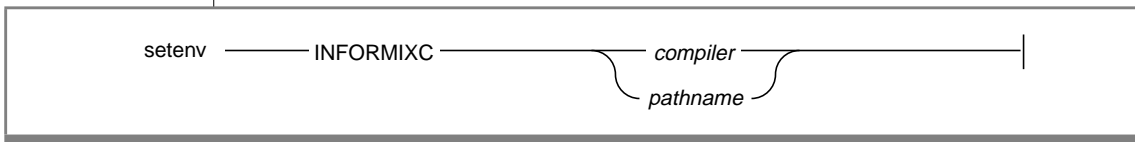
Element	Description
	A shaded option is the default action.
	Syntax within a pair of arrows indicates a subdiagram.
	The vertical line terminates the command.
	A branch below the main path indicates an optional path. (Any term on the main path is required, unless a branch can circumvent it.)
	A loop indicates a path that you can repeat. Punctuation along the top of the loop indicates the separator symbol for list items.
	A gate ($\sqrt{3}$) 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 <i>size</i> no more than three times within this statement segment.

(2 of 2)

How to Read a Command-Line Diagram

Figure 0-1 shows a command-line diagram that uses some of the elements that are listed in the previous table.

Figure 0-1
Example of a Command-Line Diagram



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

Figure 0-1 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.

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

An Answers OnLine 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 the installation insert that accompanies Answers OnLine.

Printed Manuals

To order printed manuals, call 1-800-331-1763 or send email to moreinfo@informix.com. Please provide the following information when you place your order:

- 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 of the Informix error messages and their corrective actions. For a detailed description of these error messages, refer to *Informix Error Messages in Answers OnLine*.

To read the error messages under UNIX, you can use the following commands.

UNIX

Command	Description
finderr	Displays error messages on line
rofferr	Formats error messages for printing



WIN NT

To read error messages and corrective actions under Windows NT, use the **Informix Find Error** utility. To display this utility, choose **Start→Programs→Informix** from the Task Bar. ◆

Documentation Notes, Release Notes, Machine Notes

In addition to printed documentation, the following sections describe the on-line files that supplement the information in this manual. Please examine these files before you begin using your database server. They contain vital information about application and performance issues.

UNIX

On UNIX platforms, the following on-line files appear in the `$INFORMIXDIR/release/en_us/0333` directory.

On-Line File	Purpose
<code>TFMANDOC_7.3</code>	The documentation-notes file for your version of this manual describes features that are not covered in this manual or that have been modified since publication.
<code>SERVERS_7.3</code>	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.
<code>IDS_7.3</code>	The machine-notes file describes any special actions that are required to configure and use Informix products on your computer. Machine Notes are named for the product described.

**WIN NT**

The following items appear in the Informix folder. To display this folder, choose **Start→Programs→Informix** from the Task Bar.

Item	Description
<code>doc_db.txt</code>	This item includes additions or corrections to manuals, along with information about features that may not be covered in the manuals or that have been modified since publication.
<code>rel_db.txt</code>	This item 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.

Machine notes do not apply to Windows NT platforms. ◆

Related Reading

The auditing facility provided with Dynamic 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”). The U.S. Department of Defense publishes these books.

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)

The following prominent books are about the Windows NT operating system:

- *Inside Windows NT* by Helen Custer (Microsoft Press, 1993)
- *Windows NT Administration* by Marshall Brain and Shay Woodard (Prentice-Hall, 1994)

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. In addition, many features of Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL CAE (common applications environment) standards.

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:

650-926-6571

We appreciate your feedback.

What Is Auditing?

Secure-Auditing Facility	1-3
Audit Events	1-4
Audit Masks	1-4
Auditing Process	1-7
Audit Trail	1-8
Administrative Roles	1-9
Audit Masks and Audit Instructions	1-10
User Masks	1-10
Making a Mask for User informix	1-11
Template Masks	1-11
Setting the Audit Instructions	1-12
Resource and Performance Implications	1-13
Suggested Minimum Set of Events to Audit	1-14
Special Auditing Considerations	1-15
Level of Auditing Granularity	1-15
Use of Various Masks	1-16
Audit Configuration	1-17
Turning Auditing On or Off	1-17
Specifying Types of Auditing	1-18
Auditing Modes on UNIX Systems	1-18
Auditing Modes on Windows NT Systems	1-19
Properties of Audit Files on UNIX Systems	1-19
Locating Audit Files	1-20
Starting New Audit Files	1-20
Naming Audit Files	1-21
Windows NT Event Log	1-21
Windows NT Message Server	1-22

Access to the Audit Trail.	1-22
Controlling Access to Audit Files on UNIX Systems	1-23
Controlling Access to Audit Records on Windows NT Systems	1-23
Error Modes for Writing to an Audit File or the Event Log	1-23
Halt Error Modes	1-24
Continue Error Mode	1-24
Audit Configuration and the ADTCFG File	1-24
Access to the Audit Trail.	1-25
Audit Analysis	1-27
The Importance of Audit Analysis	1-27
Preparing for Audit Analysis	1-28
Strategies for Audit Analysis	1-30
Event Failure	1-30
Event Success	1-30
Insider Attack	1-31
Browsing.	1-31
Aggregation	1-31
Responding to Identified Security Problems	1-32
DBMS Security Threats	1-33
Primary Threats.	1-33
Privileged Activity Threats	1-34
Database Server Administrator	1-34
Database System Security Officer (DBSSO)	1-34
Operating-System Administrator (OSA)	1-34
Audit Analysis Officer (AAO)	1-34
Shared-Memory Connection Threats on UNIX Systems	1-35
Introduced Malicious Software Threats	1-35
Remote-Access Threats	1-36
Obsolete-User Threats	1-36
Untrusted Software Used in a Privileged Environment	1-37
Distributed-Databases Configuration Threats	1-37

T

his chapter provides an overview of Informix Dynamic 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.

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 to reconstruct a database. Dynamic Server has archiving and backup facilities for that purpose. The [Backup and Restore Guide](#) explains these facilities.



Audit Events

Any database server activity that could potentially alter or reveal data or the auditing configuration is considered an *event*. The Dynamic 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*, which is a four-letter abbreviation. The events that you can audit with the 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 Dynamic Server is based on the notion of audit events and audit masks.

Audit Masks

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

Before you use auditing, you need to specify which audit events to audit. In other words, you need to add events to the masks. You also need to perform other tasks, which [Chapter 2, “Audit Administration,”](#) describes.

Dynamic Server has no provisions for auditing based on objects or processes. For example, you cannot ask Dynamic 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 1-10](#).

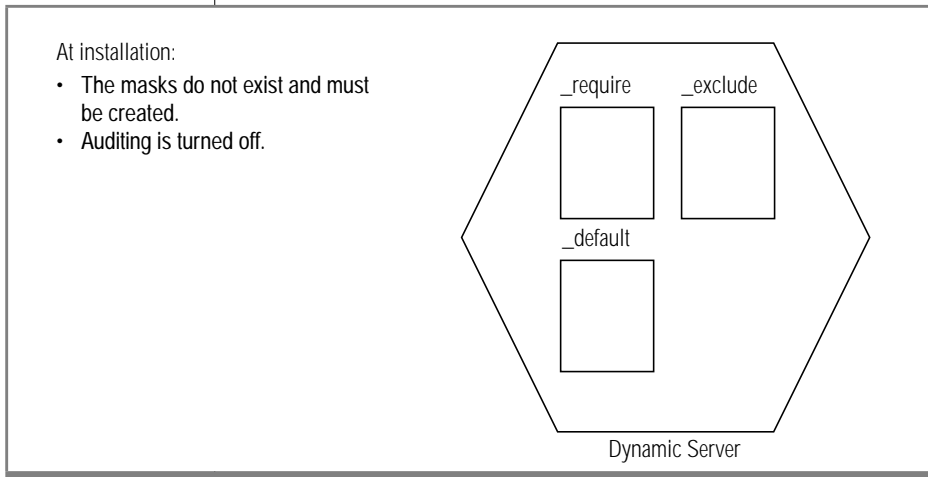


Figure 1-1
Audit Masks After Installation

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

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

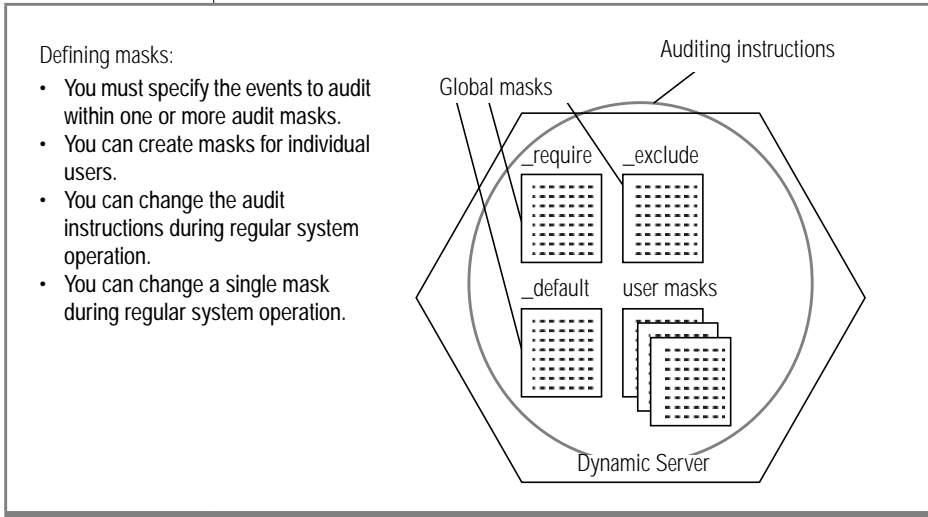
In addition to the three masks that Figure 1-1 shows, you can specify *user masks* for individual users. User masks allow you to audit some users more than others and target different types of activities for different users. A user (other than the audit administrator) cannot tell which events are being audited. User masks are described on [page 1-10](#).

You can also create *template masks*. These masks are used as a template to create new user masks. Template masks are described on [page 1-11](#).



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

Figure 1-2
The Auditing Instructions



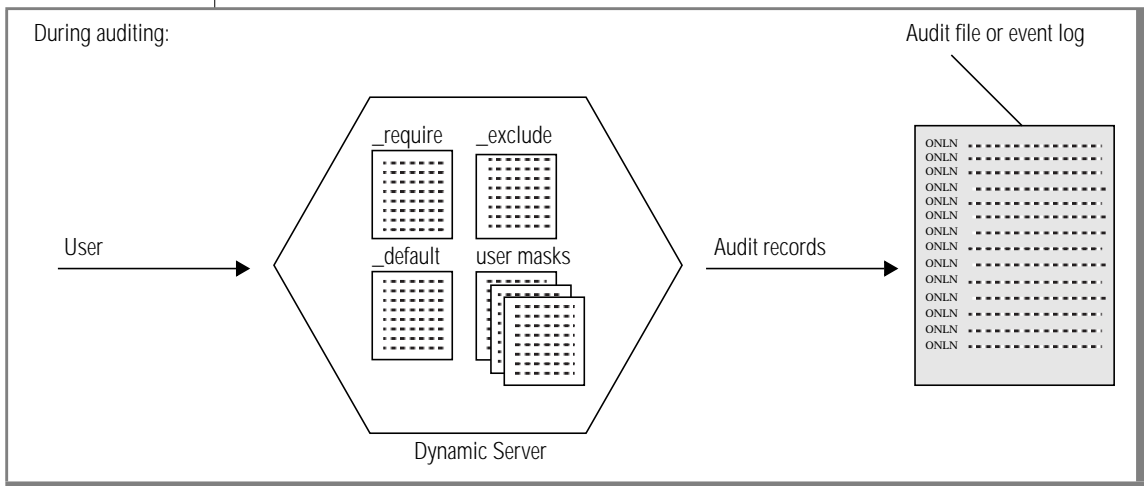
Once you define the auditing instructions and turn on auditing, you can modify one or more audit masks as needs change and potential security threats are identified. For information on how to change audit masks, see [Chapter 2, “Audit Administration.”](#)

Auditing Process

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

If you use database-server-managed auditing, the database server stores the audit records in a UNIX file called an *audit file* or in the Windows NT *event log*, as Figure 1-3 shows. The collection of audit records makes up the *audit trail*. (On a UNIX system, the audit trail might consist of more than one audit file.) When operating-system-managed auditing is used on a UNIX system, 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
- How the database server behaves if it encounters an error when writing audit records to the audit trail
- For a UNIX system, the directory in which the audit trail is located
- For a UNIX system, the maximum size of an audit file before the database server or operating system automatically starts another audit file ♦

Each of these topics is explained in [“Audit Configuration” on page 1-17](#).

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

Audit Trail

Review the audit trail regularly. Dynamic 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 1-27](#).

Administrative Roles

You can set up the following three administrative roles for the database server, in addition to any administrative roles that your operating system might have:

- The database server administrator maintains and tunes the database 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 the database server, decides at installation time whether to have separate DBSSO and/or AAO administrative roles and who should perform them.

UNIX

On a UNIX system, 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. ♦

WIN NT

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

For detailed information on roles and role separation, see [Chapter 2](#).

Audit Masks and Audit Instructions

As described in “[Audit Masks](#)” on page 1-4, an *audit mask* specifies a set of events to be audited when a user performs them. 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 following table lists four types of audit masks.

Mask Type	Mask Name
Individual user masks	<i>username</i>
Default mask	_default
Compulsory masks	_require and _exclude
Template masks	<i>_maskname</i>

The first three kinds of masks are described in “[User Masks](#)”. Template masks are described on [page 1-11](#).

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, the database server checks whether an individual user mask exists with the same *username* as the account that the user uses. If an individual user mask exists, the audit instructions in it are read first, and the **_default** mask is ignored. If no individual user mask exists, the database server reads and applies the audit instructions in the **_default** mask to that user.

In addition to default and individual masks, the database server reads and applies the audit instructions in the **_require** and **_exclude** masks. These masks are *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**, and **_exclude**.*

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

When the database server is installed, no audit masks exist. The audit administrator must specify even the default mask and the compulsory masks.

Making a Mask for User *informix*

Important: *Actions that the database server, audit administrator or user **informix** generally performs are potentially dangerous to the security of the database server. To reduce the risk of an unscrupulous user abusing the **informix** account, 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 (_). These naming rules distinguish templates from individual user masks.

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

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

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

Setting the Audit Instructions

The audit administrator sets the audit instructions that the database 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 database 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 that the database server uses are stored in the **sysaudit** table in the **sysmaster** database. The masks are updated automatically when the database server is upgraded to a newer version. Although information stored in the **sysmaster** database is available through SQL, you should use the **onaudit** utility for all audit-mask creation and maintenance. (See [Chapter 4, “Utility Syntax.”](#)) Also, see the discussion of the **sysmaster** database in your *Administrator’s Guide*.

Resource and Performance Implications

The amount of database server auditing enabled at any given time has a direct effect on operating-system resources and database server performance. All audit data that the database server generates 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 database server security-related events that all users perform. 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 database 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 database 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 database server audit-record generation can adversely affect database server performance and resources, it is still advisable to perform more than minimal auditing. Audit enough events to detect security violations and attempts to circumvent security mechanisms. This section discusses some of the points to remember when you balance security needs with the performance and resource effects of different audit levels.

Informix recommends that you audit the following events for all standard database 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 database server users. Auditing all Open operations indicates the general area within the database server where users are looking. Auditing these operations should not significantly affect database server performance; these operations are performed infrequently compared with other operations.

Creative attempts to circumvent the database server security policy are virtually impossible to detect if minimal or no auditing is performed for regular database server users. If a security violation is suspected, or if a particular user exhibits unusual behavior (as revealed by the database 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

The Dynamic 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](#)” on page A-6.

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

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

For more information on the corresponding SQL statements, see the [Informix Guide to SQL: Syntax](#).

Use of Various Masks

The **_require** mask can be a valuable tool; every database server user 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 that you specify in the **_exclude** mask are excluded from auditing. This exclusion is true of every event, including those specified in the **_require** mask. The Read Row audit event, for example, is a good candidate for the **_exclude** mask. Read Row is a common event that can generate huge amounts of potentially useless data in the audit trail.

The way that 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 auditing error modes
- Using the **ADTCFG** file
- On UNIX systems, specifying database-server-managed auditing or operating-system-managed auditing
- On UNIX systems, determining properties of the audit files ♦

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 the database server is installed.

As described in [Chapter 2](#), the AAO can turn auditing on and off at any time. The database 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 the database server is in on-line mode, existing sessions resume producing audit records.

Specifying Types of Auditing

When you turn auditing on, you can set the ADTMODE parameter in the ADTCFG file to specify the type and level of auditing.

The following subsections briefly describe the types of auditing on UNIX and Windows NT systems. For details, see [“Changing the Auditing Configuration” on page 4-14](#) and [Appendix B, “The ADTCFG File.”](#) For more information on auditing administration, see [“Administrative Roles and Role Separation” on page 2-3.](#)

UNIX

Auditing Modes on UNIX Systems

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

For example, if you set the ADTMODE configuration parameter to 1 in your ADTCFG file on a UNIX system, database-server-managed auditing is turned on automatically when the database 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 or the database server administrator or both when you turn on auditing, audit records are generated for all events that are executed by the specified roles.)

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

The OSA 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 the database server to use an audit trail that the operating system manages:

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

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

WIN NT

Auditing Modes on Windows NT Systems

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

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

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

UNIX

Properties of Audit Files on UNIX Systems

As described in [“Auditing Process” on page 1-7](#), with database-server-managed auditing on a UNIX system, the database server writes audit records to audit files in an audit trail. This section describes the audit files in more detail.

Locating Audit Files

The audit files are located in a directory that you specify with the **onaudit** utility or the ADTPATH parameter in the **SINFORMIXDIR/aaodir/adtcfg** UNIX 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 enables you to change the directory when the database 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, the database server appends it to the current audit file. If you bring the database server out of on-line mode and then put it back, the database server continues to use the same audit file. The database server starts a new audit file only under the following conditions:

- When the file reaches a specified size
- When you manually direct the database server to start a new audit file, as described in [Chapter 2](#)
- If you start database-server-managed auditing

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

WIN NT

Windows NT Event Log

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

Windows NT provides a secure event log, so the database server does not need to provide any additional security. The security log and system log are for use only by Windows NT services that are running under the **LocalSystem** user account and the Windows NT security subsystem.

Any workstation in the Windows NT network can view the event log as long as the user has sufficient access privileges. The security log is accessible only to users who belong to the Windows NT **Administrator** group, including domain administrators.

Any messages that the database server writes to the server log file, it also writes to the Windows NT event log. The database server also writes auditing records to the event log.

For information about working with events, see the [Informix Enterprise Command Center User Guide](#).

WIN NT

Windows NT Message Server

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

Because the Windows NT security log and system log are for use only by services running under the **LocalSystem** user account, Dynamic Server includes a **Message Server** service that runs under that account. The **Message Server** service communicates with the database server through the named pipes interprocess communications mechanism to receive information and write it to the event log.

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

Access to the Audit Trail

The audit trail (UNIX audit files or Windows NT event log) should be accessed only with the **onshowaudit** utility, which has its own protection, as follows:

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

UNIX

WIN NT

UNIX

Controlling Access to Audit Files on UNIX Systems

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: **informix**
Group ID: same as **\$INFORMIXDIR/aaodir**
Permissions: **660**



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

WIN NT

Controlling Access to Audit Records on Windows NT Systems

Standard users should not be able to view or alter audit records. The following characteristics control access to the event log and protect it from accidental viewing or deletion:

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

Error Modes for Writing to an Audit File or the Event Log

If the database server encounters an error when it writes to the audit file or event log, it can behave in various ways called *error modes*. You can change the error mode, as [Chapter 2](#) describes, at any time during database server operation, even after an error occurs. See the discussion of **onaudit** error modes in [Chapter 4](#).

Halt Error Modes

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

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

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

Continue Error Mode

When the database server is in *continue* error mode (0), it allows the session that received the error 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. The database server writes an error to the message log stating that an error made while writing an audit record has occurred.

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

Audit Configuration and the ADTCFG File

Parameters in the ADTCFG file represent and record the properties of the audit configuration. These parameters are ADTERR and ADTMODE, and also ADTPATH and ADTSIZE for UNIX.

The path name for the ADTCFG file follows.

Platform	ADTCFG Path Name
UNIX	<code>SINFORMIXDIR/aaodir/adtcfg</code>
Windows NT	<code>%INFORMIXDIR%\aaodir\adtcfg</code>

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 [Chapter 2](#) describes, 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 your *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 database server administrator to start an instance of the database server with invalid audit parameters. For details on how to use the **onaudit** utility to configure the ADTCFG file, see [Chapter 4](#).

Access to the Audit Trail

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

UNIX

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

aaodir

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

aaodir/adtcfg.std

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

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

aaodir

Ownership: **informix**
Group ID: **informix**
Permissions: **770**

aaodir/adtcfg.std

Ownership: **\$AAOOWNER**
Group ID: **<ao_group>**
Permissions: **664**



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

WIN NT

The following examples show how to control access to the Windows NT event log:

aaodir

Ownership: **informix**
Group ID: **Administrator**

aaodir\adtcfg.std

Ownership: **database server administrator**
Group ID: **Administrator**

♦

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

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

***Tip:** 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 appears in the security audit trail to ensure that the audit facility is operating properly.



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

Preparing for Audit Analysis

This section describes two methods to analyze database 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 the database server manages. You can then use SQL to generate reports based on this data. With the SQL-based method, you can create and use customized forms and reports to manipulate and selectively view audit data, which provides a flexible and powerful audit-analysis procedure. However, be sure that records are not deleted or modified from either the intermediate file or the database prior to analysis.

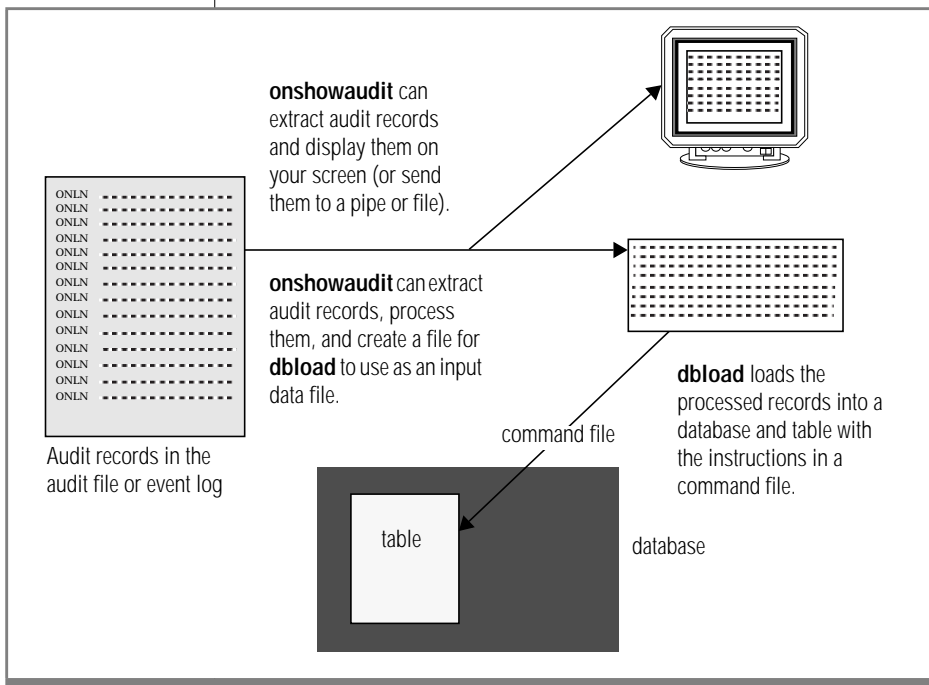


Important: *The SQL-based procedure is more convenient but remains untrusted because users can use SQL data-manipulation statements to tamper with the records that are copied into a table.*

Both methods rely on a utility called **onshowaudit**, which [Chapter 3](#) discusses and [Chapter 4](#) describes. For either method, you can extract audit events for specific users and/or database servers.

[Figure 1-4 on page 1-29](#) shows the preparation process for both analysis methods. [Chapter 3](#) explains each step in detail.

Figure 1-4
Preparing for Audit Analysis



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

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

- The operating system supports auditing.
- The database server supports operating-system auditing on this platform.
- For records in the operating-system audit trail, your database server must be registered as a protected subsystem with your operating system, as described in the UNIX machine-notes file. (See “[Documentation Notes, Release Notes, Machine Notes](#)” on page 12 of the Introduction.)

UNIX

- Database server users have performed activities that generated audit records.
- Operating-system auditing is on. ♦

On a Windows NT system, users can clear or remove audit logs by using the Windows NT Event Viewer administrative tool. ♦

Strategies for Audit Analysis

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

Event Failure

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

Event Success

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

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

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

Insider Attack

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

Browsing

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](#).

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, database 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 permissions or access privileges.

Privileged Activity Threats

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

Database Server Administrator

The database server administrator controls and monitors the database. The countermeasure to a threat from the database server administrator is independent scrutiny of the DBMS audit trail because auditing all database 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 on UNIX Systems

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

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

The path name for the UNIX configuration file is **\$INFORMIXDIR/etc/sqlhosts**.

Introduced Malicious Software Threats

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

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

Make all users aware of the dangers of executing software of unknown or untrusted origin. Further, 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 of having remote access to privileged data when the user grants another user direct access to that data. This situation could lead to data that is inappropriately accessed remotely.

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

Obsolete-User Threats

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

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

To avoid this problem, have the OSA notify the database server administrator when a user account is removed from the operating system. The database 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 database server administrators execute untrusted software. This untrusted software could use the privileges of the database 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 database administrators execute 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. ♦

UNIX

Audit Administration

Administrative Roles and Role Separation	2-3
The Database Server Administrator	2-4
The Database System Security Officer	2-4
The Audit Analysis Officer	2-5
Other Administrative Roles and Users	2-6
The Database Administrator	2-6
The Operating-System Administrator	2-6
System Users	2-6
Privileged Users	2-7
Role Separation	2-7
Assigning Roles	2-7
Configuring and Enforcing Role Separation	2-8
Setting Up Auditing	2-10
Setting Up the Default and Compulsory Masks	2-10
Specifying a UNIX Directory for the Audit Trail	2-11
Setting the Error Mode	2-11
Setting the Audit Level	2-12
Activating Auditing	2-13
Maintaining Audit Masks	2-14
Creating Audit Masks	2-14
Creating a Template Mask	2-15
Creating a User Mask from a Template Mask	2-15
Creating a User Mask Without a Template Mask	2-16
Adding One or More Masks Using an Input File	2-16
Displaying Audit Masks	2-17
Modifying Audit Masks	2-18
Deleting Audit Masks	2-19

Maintaining the Audit Configuration	2-19
Displaying the Audit Configuration.	2-20
Starting a New Audit File on a UNIX System	2-21
Changing the Audit Mode on a UNIX System	2-22
Changing the Audit Mode on a Windows NT System.	2-22
Changing the Audit Error Mode	2-23
Turning Off Auditing.	2-23

This chapter explains how to set up and administer auditing on your database server, provided that the database server is installed and functions properly. This chapter discusses the following topics:

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

Administrative Roles and Role Separation

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

- The database server administrator
- 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. It tells how to set up role separation and provides guidelines for how to assign roles.

The Database Server Administrator

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

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

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

Tip: A database server administrator is any user who belongs to the group **informix** (UNIX) or logs on as user **informix** (Windows NT), 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 on a UNIX system is any user who belongs to the group that owns **SINFORMIXDIR/dbssodir**. On a Windows NT system, the administrator uses registry settings to specify DBSSO users.





Important: The **onaudit** utility can create a potential threat to the security of the database server. An unscrupulous user can abuse a DBSSO account; for example, 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 can specify whether and how auditing is enabled, how the system responds to error conditions, and who is responsible for managing the audit trail.

UNIX

For database-server-managed auditing on a UNIX system, the AAO also determines the directory for the audit trail and the maximum size of each audit file. For operating-system-managed auditing on a UNIX system, 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 database server session, the new values are written to the **adtcfg.servernum** file for that instance of the database server.

The installation script for the database server creates a **\$INFORMIXDIR/aaodir** UNIX directory or a **%INFORMIXDIR%\aaodir** Windows NT 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 UNIX permissions or Windows NT access privileges to view all the data in the database server to analyze events that might involve sensitive information. The AAO decides whether to audit all actions of the DBSSO and the database server administrator.

Tip: An AAO on a UNIX system is any user who belongs to the group that owns **\$INFORMIXDIR/aaodir**. On a Windows NT system, the administrator uses registry settings to specify AAO users.



Other Administrative Roles and Users

A number of other, more minor, roles might be involved in database 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 database server roles and users, see your [Administrator's Guide](#).

The Operating-System Administrator

The OSA carries out responsibilities and tasks that the database server requires from the operating system. The OSA enables role separation, grants and revokes access to and from the database server if role separation is enforced, and adds new AAO, DBSSO, and database 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 the database server, and no special database server protection mechanisms are associated with OSA tasks. For more information, refer to your operating-system documentation.

System Users

All operating-system accounts, including those for the database server administrator, DBSSO, AAO, and the account called **informix**, potentially can use the database server. All users with accounts who want to use the database server must explicitly be granted access to the database server if role separation is configured to enforce access control on database server users. 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 the database server recognizes as having additional privileges and/or responsibilities. These privileged users include the database 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 database server option that allows users to perform different administrative tasks. Role separation is based on the principle of separation of duties, which reduces security risks with a checks-and-balances mechanism in the system. For example, the person who determines what to audit (DBSSO) should be different 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 (the database 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 database 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 database server administrator and DBSSO accounts as possible.

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

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

Configuring and Enforcing Role Separation

Role separation is configured during database server installation. The database server administrator, or the person who installs the database server, enforces role separation and decides which users will be the DBSSO and AAO. To find the group of the database server administrator, DBSSO, and AAO, you must look at the appropriate directory of `$INFORMIXDIR` on a UNIX system or `%INFORMIXDIR%` on a Windows NT system.

UNIX

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

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

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

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

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

```
total 14
drwxrwxr-- 2 informix informix 512 Nov 21 09:56 aaodir/
drwxr-xr-x 2 informix informix 1536 Nov 30 18:35 bin/
drwxrwxr-x 2 informix informix 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_dbss**, and the database 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 group setting means that all users are allowed to connect to the database server. If the file contains a specific name such as **ixusers=engineer**, then only members of the group **engineer** can gain access to the database server. ♦

WIN NT

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

For information on environment variables, see the [Informix Guide to SQL: Reference](#). For information on configuring role separation, see your [Administrator's Guide](#).

Setting Up Auditing

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

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

UNIX

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. Also, be sure that you understand which auditing events to place in which masks. For details, see [Chapter 1, “What Is Auditing?”](#)

Use the **onaudit** utility to add audit events to audit masks. The audit events and their mnemonics are listed in [Appendix A, “Audit Events.”](#) [Chapter 4](#) shows the complete syntax for **onaudit**.

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

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

You can add audit events to the `_require` and `_exclude` masks in the same way. For specifics, see [Chapter 4](#).

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

Specifying a UNIX Directory for the Audit Trail

As the AAO, when you turn on auditing on a UNIX system, you specify that either the database 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 the database 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. For more information, see [Chapter 4](#). You can also set up the type of auditing and specify the directory with the ADTCFG file that [Appendix B](#) shows.

Setting the Error Mode

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

Use the `onaudit` utility or the ADTCFG file to set the error mode. For the `onaudit` syntax, see [Chapter 4](#). For the ADTERR configuration parameter, see [Appendix B](#).

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

```
onaudit -e 0
```

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

```
onaudit -e 3
```

Setting the Audit Level

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

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

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

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

To ensure that the appropriate database server activities are monitored, review the audit records that are stored in the operating-system audit trail, database server audit files, or Windows NT event log. The database 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 database server audit-record generation might have a negative effect on database server performance and resources, it nevertheless is advisable to perform more than the minimal database server audit. This additional audit improves the likelihood that you will detect security violations and any attempts to circumvent security mechanisms.*

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

Tip: You can specify disk space using the Windows NT Event Viewer administration tool.

For complete information on database server performance considerations, see your [Performance Guide](#).



Activating Auditing

Auditing is turned off by default when you install the database 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 the database server is initialized.

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

```
onaudit -l 1
```

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

The AAO can configure Dynamic Server to turn auditing on when shared memory is initialized when the ADTMODE parameter is set to a number from 1 through 8 (UNIX) or to 1, 3, 5, or 7 (Windows NT) in the ADTCFG file. For details on ADTMODE parameter values, see [“Changing the Auditing Configuration” on page 4-14 and Appendix B](#).

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

To turn off auditing after it starts, see [“Turning Off Auditing” on page 2-23](#).



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 the database 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 database server or your operating system administers the audit records.

Creating Audit Masks

You can create masks that more closely match the types of activities that individual users perform than do default and 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 mask, 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 mask 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
```

For the syntax for creating a user mask and another example, see [Chapter 4](#).

Adding One or More Masks Using an Input File

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

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

Figure 2-2 shows 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** UNIX directory or in the **%INFORMIXDIR%\aaodir** Windows NT 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 1-24.](#)) 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](#)”. 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.)

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. For the syntax and another example of how to modify a mask, see [Chapter 4](#).

Deleting Audit Masks

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

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

For the syntax of the **onaudit** utility, see [Chapter 4](#).

Maintaining the Audit Configuration

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

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

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

Displaying the Audit Configuration

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

UNIX

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

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright (c) Informix Software, Inc., 1998

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 database-server-managed auditing is on.
- ADTERR is set to 0, which indicates a continue error mode.
- ADTPATH shows the default directory for audit files.
- ADTSIZE, which represents the maximum size of the audit file, is specified as 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) **sysadinfo** table in the **sysmaster** database. For details, see your [Administrator's Guide](#). ♦

WIN NT

Figure 2-3 shows sample output from the **onaudit -c** command on a Windows NT system.

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright (c) Informix Software, Inc., 1998

Current audit system configuration:
  ADTMODE = 1
  ADTERR = 0
```

Figure 2-4
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 database-server-managed auditing is on.
- ADTERR is set to 0, which indicates a continue error mode. ♦

UNIX

Starting a New Audit File on a UNIX System

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

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

```
onaudit -s 20000
```

- Use **onaudit -n** to start a new audit file without changing the maximum size. This option, 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 the database 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 that you start database-server-managed auditing.

UNIX

You can use more than one flag at a time in an **onaudit** command. For the **onaudit** utility syntax to start a new audit file, change the audit-file size, or change the pathname of the audit directory, see [Chapter 4](#).

Changing the Audit Mode on a UNIX System

On a UNIX system, use the **onaudit** utility to change between operating-system-managed auditing and database-server-managed auditing and to change the mandatory auditing of the database server administrator or DBSSO or both.

For example, to start basic operating-system-managed auditing, enter the following command:

```
onaudit -l 2
```

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

```
onaudit -l 8
```

WIN NT

Changing the Audit Mode on a Windows NT System

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

```
onaudit -l 1
```

To start auditing and automatically audit the actions of the database server administrator, enter the following command:

```
onaudit -l 5
```

Changing the Audit Error Mode

As explained in [Chapter 1](#) and in “[Setting the Error Mode](#)” on page 2-11, the database server behaves in one of three ways if it encounters an error when it writes to the current UNIX audit file or to the Windows NT event log. You can change the audit error mode using the **onaudit** utility. The following example directs the database server to suspend processing of the current thread and continue the write attempt until it succeeds:

```
onaudit -e 1
```

Turning Off Auditing

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

```
onaudit -l 0
```



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

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

Audit Analysis

The Audit-Record Format	3-3
Audit Analysis Without SQL	3-5
Audit Analysis with SQL.	3-6
Preparing for SQL Audit Analysis	3-6
Creating a Data File for dbload	3-6
Creating a Database and Table for Audit Data	3-7
Create a Database	3-7
Create a Table	3-7
Privileges to Protect Audit Data.	3-9
Creating a Command File for dbload	3-9
Loading Audit Data into a Database	3-10
SQL Audit-Analysis Considerations	3-10

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

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

This chapter applies whether you use the database server or your operating system to store and maintain the audit trail. An overview of the audit analysis process is in [Chapter 1, “What Is Auditing?”](#)

The Audit-Record Format

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

- The first part is an operating-system audit header, if operating-system auditing is used on a UNIX system. The audit header contains information that the operating system supplies. ♦
- The second part of the audit record is generated by the database server.

[Figure 3-1 on page 3-4](#) shows the format of the database server audit records.

Figure 3-1
Audit-Record Format

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 Dynamic Server events.

date and time indicates when the audit event was recorded.

hostname is the name of the UNIX host computer of the client that executes the audit event.

hostname.domain.ext is the name of the Windows NT host computer, domain, and extension of the client executing the audit event.

pid is the process ID of the client program that causes the database 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 database server audit event that the user executed, for example, ALFR (Alter Fragment).
additional fields	indicates any additional fields used to identify databases, tables, and so on. (In a sense, the database 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** or Windows NT utilities to analyze it. You can also choose to put the data in a database and analyze it with SQL, as the next section describes.

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

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

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

The command-line syntax for how to extract information with **onshowaudit** is explained in [Chapter 4, “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 and 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 in an output file. The following example extracts audit records for the user **pat** from the database-server-managed audit file **laurel.11** and directs the records to the **records_pat** output file:

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

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

The command-line syntax to extract information with **onshowaudit** is explained in [Chapter 4](#).



Creating a Database and Table for Audit Data

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

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 or access privileges makes it attractive in a secure environment. For more information about UNIX permissions or Windows NT access privileges, refer to [“Privileges to Protect Audit Data” on page 3-9](#).

The following SQL statement creates an ANSI-compliant database:

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

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 same order as the example in [Figure 3-2 on page 3-8](#). The sample schema reflects the format of the **dbload** data file created by **onshowaudit**.

The sample CREATE TABLE statement that [Figure 3-2](#) shows creates an audit table with the name **frag_logs**. It has **objname** and **partno** fields that store fragmentation information from the event records. Audit tables created with Dynamic Server releases prior to Version 7.0 do not have these two fields.

```
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 that the statement in Figure 3-2 creates does not have any indexes. To improve audit-analysis performance, you can place indexes on columns within the table, depending on the type of analysis that you perform. For guidance on indexing columns, see your [Performance Guide](#).

In the audit table, the **objname** value (object name) consists of the **idxname** value (index name) and **tablename** value (table name), or another, similar identifier. The **partno** value consists of the partition number or **frag_id** value, which is always the physical partition of the record, and the **oldpartno** value (old partition number). Additionally, the **row_num** value represents the row number, which is always the physical row ID for the record being manipulated, **rowid** value (row identifier), and **oldrowid** value (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 or access privileges prevent access to the tables, proper security practice protects the audit-analysis table in a database that is not ANSI-compliant by revoking access from all other users as soon as that table is created.

You can use the following SQL statements to control access:

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

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

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

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

Creating a Command File for dbload

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

Loading Audit Data into a 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 3-7](#):

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

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

SQL Audit-Analysis Considerations

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

To perform audit analysis with SQL, you must use a program to access the database and table that you created. Use the DB-Access utility to construct and execute SQL statements or develop an application with an 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 the database server, you might increase the scope of your collection of audit events to pinpoint the problem. If you feel certain you have a security problem, see [Chapter 1](#).

Utility Syntax

The onaudit Utility	4-4
Showing Audit Masks	4-5
Creating or Adding an Audit Mask	4-6
The Audit-Mask Specification	4-7
The onaudit Input-File Format	4-9
Modifying an Audit Mask	4-10
Deleting Audit Masks	4-11
Starting a New Audit File on a UNIX System	4-12
Storing Database-Server Audit Files	4-12
Storing Operating-System Audit Files	4-12
Showing the Auditing Configuration	4-13
Changing the Auditing Configuration	4-14
Using the -e Option	4-16
Using the -l Option	4-16
The onshowaudit Utility	4-17

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

- The **onaudit** utility performs the following operations on both UNIX and Windows NT systems:
 - Displays audit masks
 - Creates audit masks
 - Modifies audit masks
 - Deletes audit masks
 - Shows the audit configuration
 - Changes global auditing activities
 - Enables and disables auditing
 - Sets the error mode
 - Establishes mandatory auditing for various administrative roles
- The **onshowaudit** utility performs the following operations:
 - Extracts audit information from the audit trail
 - Prepares extracted audit data for **dbload** to use

UNIX

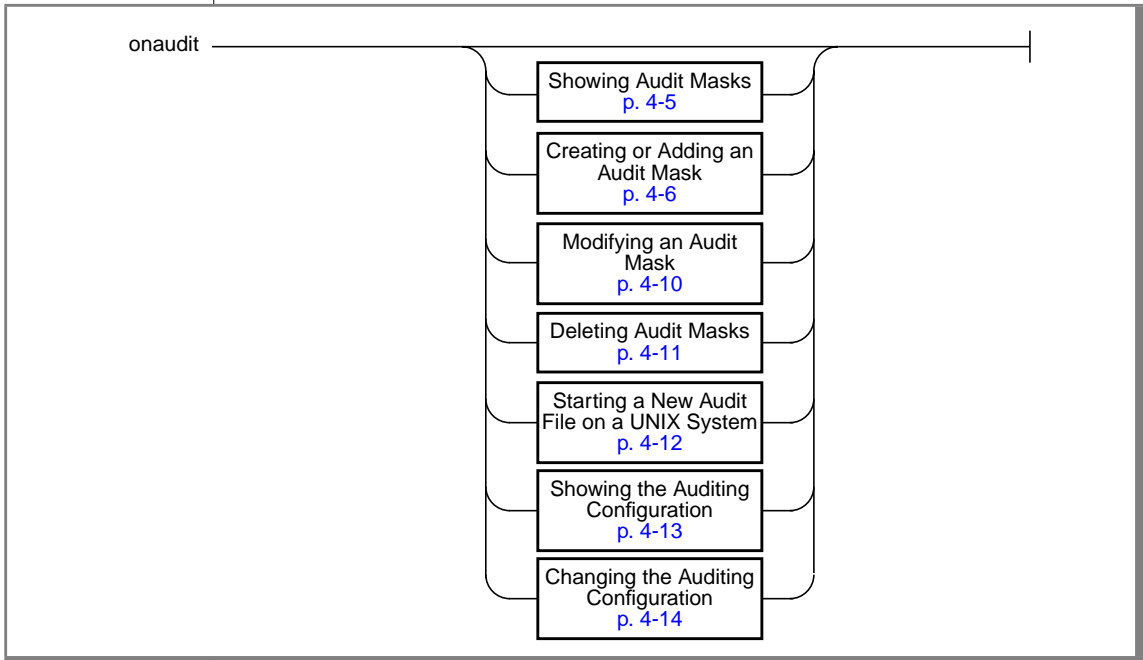
- The **onaudit** utility also performs the following operations on UNIX systems:
 - Starts a new audit file in the audit trail
 - Sets the directory in which audit files reside
 - Specifies the maximum size for each audit file
 - Determines whether the database server or the operating system manages the audit trail ♦

WIN NT

- The **onaudit** utility also performs the following operations on Windows NT systems:
 - Stores the audit-trail records in the event log
 - Establishes auditing that the database server manages ♦

The onaudit Utility

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



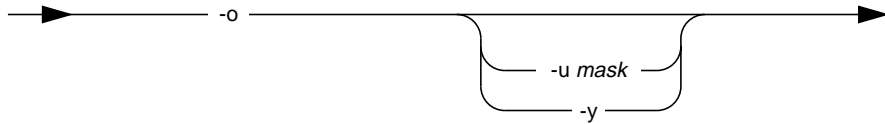
If your system has role separation, only the DBSSO or AAO can run the **onaudit** utility. The DBSSO can perform only **onaudit** functions that involve audit masks, and the AAO can perform only **onaudit** functions that involve audit configuration parameters. Without role separation, the user **informix**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

The DBSSO can change audit masks dynamically. Changes to user, default, template, and 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



Element	Purpose	Key Considerations
-o	Outputs audit masks.	None.
-u mask	Names a specific mask to display.	Additional Information: <i>Mask</i> can be any existing mask.
-y	Automatically responds <i>yes</i> 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](#).

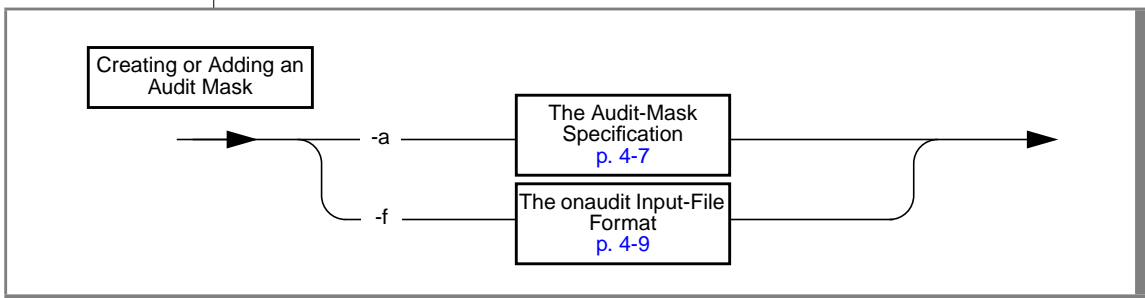
```
maskname      basemask      audit_events
```

Because the database 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.

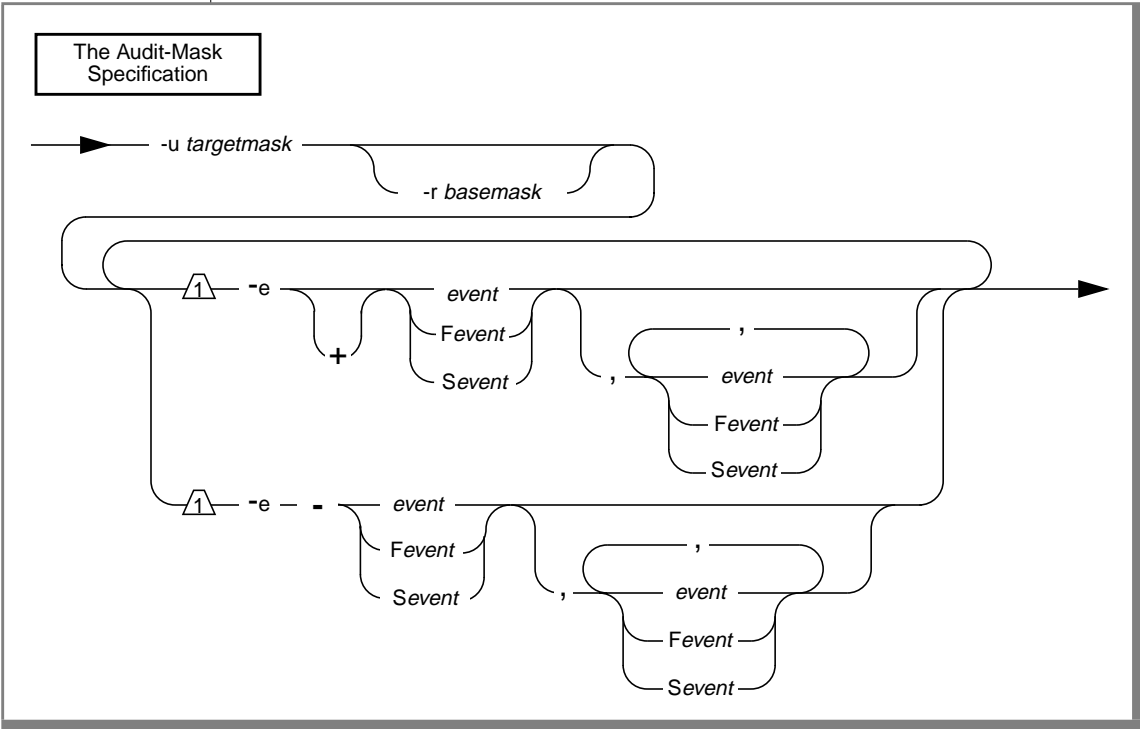
```
pat           -           LKTB,CRTB,FADCK
```

Creating or Adding an Audit Mask



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 <i>targetmask</i> .	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 <i>targetmask</i> .	None.
-e	Indicates that the arguments that follow specify audit events to be added to or removed from <i>targetmask</i> .	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 <i>targetmask</i> . The current events listed in the base mask are applied to the target mask.	Additional Information: If no <i>basemask</i> is specified and no events are specified with the -e flag, onaudit creates an empty target mask. If a <i>basemask</i> 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 <i>targetmask</i> must be eight characters or less.
Fevent	Specifies that only failed event attempts are to be audited.	Additional Information: The <i>event</i> can include the event code (mnemonic) for any event listed in the table “ Audit-Event Mnemonics ” on page A-1.
Sevent	Specifies that only successful event attempts are to be audited.	Additional Information: The <i>event</i> can include the event code (mnemonic) for any event listed in the table “ Audit-Event Mnemonics ” on page A-1.
<i>event</i>	Names an event to audit, whether the event execution succeeds or fails.	Additional Information: The <i>event</i> can include the event code (mnemonic) for any event listed in the table “ Audit-Event Mnemonics ” on page A-1.

(1 of 2)



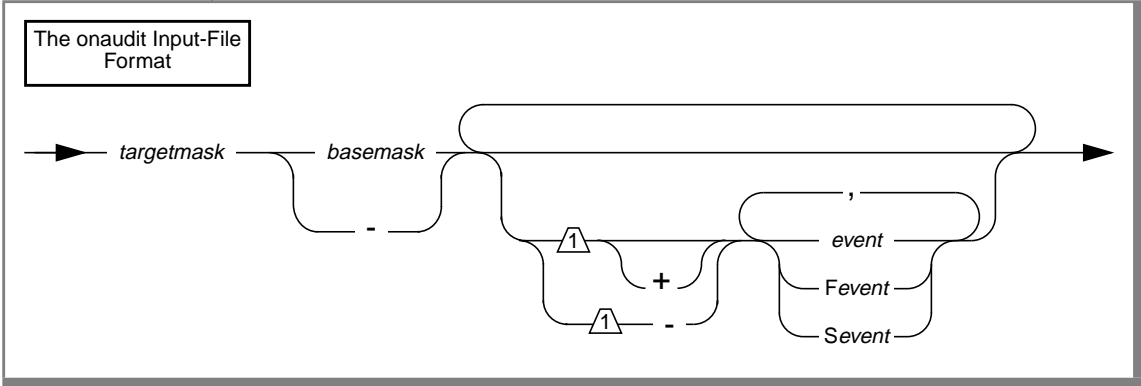
Tip: Do not include any spaces in the events list or you might get unpredictable results.

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



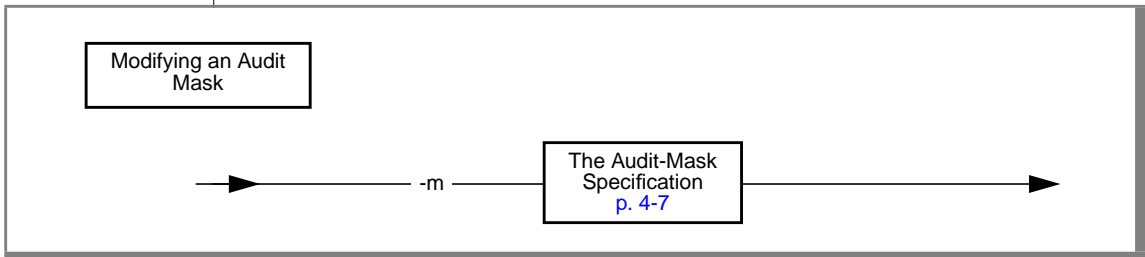
Element	Purpose	Key Considerations
+	Indicates that the events that follow are to be added to the list of audit events in <i>targetmask</i> .	None.
-	Used before an event, it indicates that the events that follow are to be removed from the list of audit events in <i>targetmask</i> . Used alone, it creates an empty mask.	None.
<i>Fevent</i>	Specifies that only failed event attempts are to be audited.	Additional Information: The <i>event</i> can include the event code (mnemonic) for any event listed in the table “ Audit-Event Mnemonics ” on page A-1.
<i>Sevent</i>	Specifies that only successful event attempts are to be audited.	Additional Information: The <i>event</i> can include the event code (mnemonic) for any event listed in the table “ Audit-Event Mnemonics ” on page A-1.
<i>basemask</i>	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.
<i>event</i>	Names an event to audit.	Additional Information: The <i>event</i> can include the event code (mnemonic) for any event listed in the table “ Audit-Event Mnemonics ” on page A-1.
<i>targetmask</i>	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



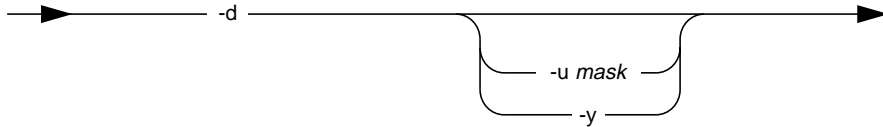
Element	Purpose	Key Considerations
-m	Modifies an existing audit mask.	None.

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



Element	Purpose	Key Considerations
<code>-d</code>	Deletes audit masks.	None.
<code>-u mask</code>	Names a specific mask to delete.	Additional Information: <i>Mask</i> can be any existing mask.
<code>-y</code>	Automatically responds <i>yes</i> 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 on a UNIX System

Starting a New Audit File

→ -n →

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

Storing Database-Server Audit Files

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

Storing Operating-System Audit Files

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

Showing the 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.

UNIX

Figure 4-1 shows sample audit-configuration output on a UNIX system.

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright (c) Informix Software, Inc., 1998

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

Figure 4-1
Sample
Audit-Configuration
Output

◆

WIN NT

Figure 4-1 on page 4-13 shows sample audit-configuration output on a Windows NT system.

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright (c) Informix Software, Inc., 1998

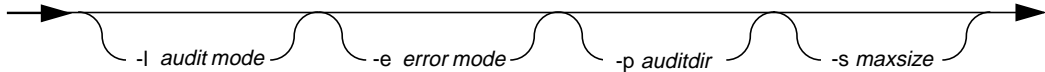
Current audit system configuration:
ADTMODE      = 1
ADTERR       = 0
```

Figure 4-2
Sample
Audit-Configuration
Output

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

Changing the Auditing Configuration

Changing the Auditing
Configuration



Element	Purpose	Key Considerations
-e error mode	Specifies the error-handling method for auditing when a record cannot be written to the audit file or event log.	<p>Restrictions: The <i>error mode</i> parameter can have one of the following values: 0, 1, 3.</p> <p>Additional Information: This option pertains to the value set for the ADTERR parameter in the ADTCFG file. The value can be changed only when auditing is on. For details of the valid <i>error mode</i> values, see “Using the -e Option” on page 4-16.</p>

(1 of 2)

Element	Purpose	Key Considerations
-l <i>audit mode</i>	Specifies the auditing mode.	<p>Restrictions: The <i>audit mode</i> parameter can have one of the following values on a UNIX system: 0, 1, 2, 3, 4, 5, 6, 7, 8. The <i>audit mode</i> parameter can have one of the following values on a Windows NT system: 0, 1, 3, 5, 7.</p> <p>Additional Information: This option pertains to the value set for the ADTMODE parameter in the ADTCFG file. For details of the valid <i>audit mode</i> values, see “Using the -l Option” on page 4-16.</p>
-p <i>auditdir</i>	Names the UNIX directory in which the database server creates audit files.	<p>Restrictions: You can change the <i>auditdir</i> value only for database-server-managed auditing.</p> <p>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. The database server starts a new audit file in the new directory, beginning with the first available number that is equal to or greater than 0.</p>
-s <i>maxsize</i>	Specifies the maximum size (in bytes) of an audit file.	<p>Restrictions: The <i>maxsize</i> can be any value between 10,240 bytes and approximately 2 gigabytes (the maximum value of a 32-bit integer). If you specify a size that is less than the minimum, it will be set automatically at the minimum. You can specify the <i>maxsize</i> value only for database-server-managed auditing.</p> <p>Additional Information: This option pertains to the value set for the ADTSIZE parameter in the ADTCFG file. When an audit file reaches or exceeds <i>maxsize</i>, the database server closes the current file and starts new audit file.</p>

(1 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 the database server is to continue processing the thread and note the error in the message log. Errors for subsequent attempts to write to the UNIX audit file or Windows NT event log are also sent to the message log.

The **-e** option has the following *halt modes*:

- 1 indicates that the database 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 the database 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*. The database 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 database-server-managed auditing for all sessions but does not automatically audit DBSSO and the database server administrator actions.
- On UNIX systems, 2 turns on operating-system-managed auditing but does not automatically audit DBSSO or database server administrator actions.
- 3 turns on database-server-managed auditing and automatically audits DBSSO actions.

- On UNIX systems, 4 turns on operating-system-managed auditing and automatically audits DBSSO actions.
- 5 turns on database-server-managed auditing and automatically audits database server administrator actions.
- On UNIX systems, 6 turns on operating-system-managed auditing and automatically audits database server administrator actions.
- 7 turns on database-server-managed auditing and automatically audits DBSSO and database server administrator actions.
- On UNIX systems, 8 turns on operating-system-managed auditing and automatically audits DBSSO and database server administrator actions.

The onshowaudit Utility

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

The records are formatted for output. By default, **onshowaudit** displays the extracted information on the screen. You can redirect the formatted output to a file or pipe and can specify that **onshowaudit** reformat the output for use with the **dbload** utility.

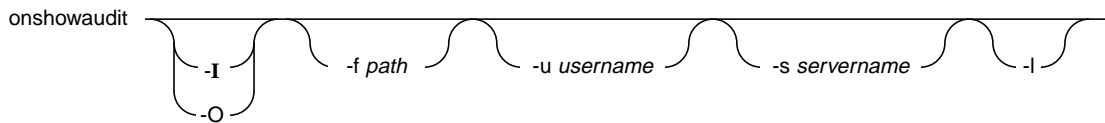
The **onshowaudit** utility extracts data from an audit trail, but it does not process the records or delete them from the audit trail.

The audit trail should be accessed only with the **onshowaudit** utility, which has its own protection:

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

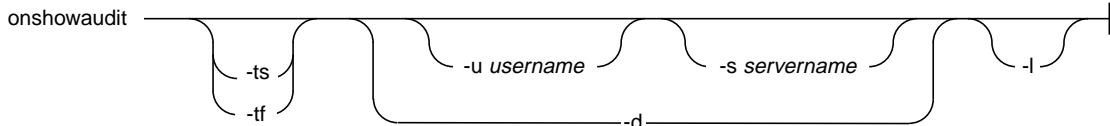
UNIX

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



WIN NT

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



Important: If you include the `-l` option in your **onshowaudit** command, you must remove the six header lines that appear in the output file before you use the file as input for **dbload**. The **dbload** utility cannot process the generated header lines.

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

Element	Purpose	Key Considerations
-d	On a Windows NT system, assumes the default values for the user (<i>current user</i>) and the database server (INFORMIXSERVER)	None.
-f path	On a UNIX system, names a specific audit trail to examine, only for database-server-managed auditing.	<p>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.</p> <p>If an <i>incomplete pathname</i> (nothing but a filename) is specified, the onshowaudit utility searches the ADTPATH directory for that file and extracts audit data from it.</p> <p>If a <i>complete pathname</i> is specified, the onshowaudit utility extracts audit data from the named file.</p>
-I	On a UNIX system, uses the Informix (database server) audit trail.	None.
-l	Directs onshowaudit to take the extracted information and reformat it for dbload to use.	References: For information on the file format, see Chapter 3, "Audit Analysis." For information on the dbload utility, see the <i>Informix Migration Guide</i> .
-O	On a UNIX system, uses the operating-system audit trail.	None.
-tf	On a Windows NT system, shows only <i>failure</i> audit records.	None.
-ts	On a Windows NT system, shows only <i>success</i> audit records.	None.
-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.

UNIX



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 on a UNIX system, **onshowaudit** calls operating-system utilities to extract from the operating-system audit trail audit records that the Informix DBMS generates.

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

***Important:** Version 7.2 and Version 7.3 of the **onshowaudit** utility can parse and process the new and updated record structures for fragmented tables and indexes (which can span multiple partitions.) Do not use **onshowaudit**, Version 7.2 or Version 7.3, to analyze records that Version 6.0 created, or you might encounter unexpected behavior. Version 7.2 and Version 7.3 of **onshowaudit** expect to find an additional field for fragmentation (**partno**) in certain audit records, but this field is absent in audit records prior to Version 7.0.*

Audit Events

This appendix contains the following two tables:

- Auditable events, listed alphabetically by event mnemonic (below)
- Audit-event records and their fields (see [page A-6](#))



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

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
ALOP	Alter Optical Cluster
ALTB	Alter Table

Audit-Event Mnemonics

Mnemonic	Event Name
BGTX	Begin Transaction
CLDB	Close Database
CMTX	Commit Transaction
CRAM	Create Audit Mask
CRBS	Create Storage Space
CRDB	Create Database
CRDS	Create Dbspace
CRIX	Create Index
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
DNDM	Disable Disk Mirroring
DRAM	Delete Audit Mask
DRBS	Drop Storage Space
DRCK	Drop Chunk
DRDB	Drop Database
DRDS	Drop Dbspace
DRIX	Drop Index

(2 of 5)

Mnemonic	Event Name
DRLG	Drop Transaction Log
DROP	Drop Optical Cluster
DRRL	Drop Role
DRSN	Drop Synonym
DRSP	Drop Stored Procedure
DRTB	Drop Table
DRTR	Drop Trigger
DRVW	Drop View
EXSP	Execute Stored Procedure
GRDB	Grant Database Access
GRFR	Grant Fragment Access
GRRL	Grant Role
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

(3 of 5)

Mnemonic	Event Name
ONMN	onmonitor
ONMO	onmode
ONPA	onparams
ONSP	onspaces
ONST	onstat
ONTP	ontape
ONUL	onunload
OPDB	Open Database
RDRW	Read Row
RLOP	Release Optical Cluster
RLTX	Rollback Transaction
RMCK	Clear Mirrored Chunks
RNDB	Rename Database
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

(4 of 5)

Mnemonic	Event Name
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
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

(5 of 5)

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 database 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, the database server creates several audit records to carry the complete information.*



Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
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 ¹⁸			owner	frag flags ¹⁵	dbspaces
Index, Alter	ALIX	dbname	tabid					owner ¹⁴	cluster flag ^{9, 14}	index name ¹⁴
Optical Cluster, Alter	ALOP	dbname			cluster size			owner		cluster name
Table, Alter	ALTB	dbname	old tabid		new tabid ¹⁴	frag_id				new part-nolist ¹⁴
Transaction, Begin	BGTX									
Database, Close	CLDB	dbname								
Transaction, Commit	CMTX									
Audit Mask, Create	CRAM							user id		
Storage Space, Create	CRBS	storage space name						owner	mirror status ¹	media
Database, Create	CRDB	dbname								dbspace name
Dbpace, Create	CRDS	dbspace name							mirror status ¹	
Index, Create	CRIX	dbname	tabid	idxname		frag_id		owner	frag flags ¹⁵	dbspacelist
Optical Cluster, Create	CROP	dbname	tabid		cluster size			owner		cluster name
Create Role	CRRL	dbname		rolename						

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Synonym, Create	CRSN	dbname	syn. tabid		base tabid			owner	syn. type ⁷	synonym name
Stored Procedure, Create	CRSP	dbname	proc. id					owner		procedure name
Table, Create	CRTB	dbname	tabid	tablename		frag_id		owner	frag flags ¹⁵	dbspacelist
Trigger, Create	CRTR	dbname	tabid				trigger id ¹⁴	owner ¹⁴		trigger name ¹⁴
View, Create	CRVW	dbname	view tabid					owner		view name
Row, Delete	DLRW	dbname	tabid		partno	frag_id	row-num ¹⁴			
Chunk, Bring Off-line	DNCK				chunk number				mirror status ¹	
Disk Mirroring, Disable	DNDM				dbspace number					
Audit Mask, Delete	DRAM							user id		
Storage Space, Drop	DRBS	storage space name								
Chunk, Drop	DRCK	dbspace name							mirror status ¹	path
Database, Drop	DRDB	dbname								dbpassword
Dbospace, Drop	DRDS	dbspace name								
Index, Drop	DRIX	dbname	tabid					owner		index name
Transaction Log, Drop	DRLG				log number					
Optical Cluster, Drop	DROP	dbname						owner		cluster name
Drop Role	DRRL	dbname		rolename						

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 ²¹	partnolist
Trigger, Drop	DRTR	dbname						trigger id		
View, Drop	DRVW	dbname	view tabid						drop-flags ²¹	
Stored Procedure, Execute	EXSP	dbname	proc. id							
Grant Database Access	GRDB	dbname			privilege ⁵					grantees ⁴
Grant Fragment Access	GRFR	dbname	tabid	fragment	privilege ^{5, 14}			grantor		grantee ^{4, 14}
Grant Role	GRRL	dbname		rolename				grantor		grantees ⁴
Grant Table Access	GRTB	dbname	tabid		privilege ^{5, 14}			grantor		grantee ^{4, 14} , update columns, select columns ^{4, 14}
Row, Insert	INRW	dbname	tabid			partno	frag_id	rowid		
Database Log Mode, Change	LGDB	dbname							log status ⁶	
Table, Lock	LKTB	dbname	tabid						lock mode ⁸	
Audit Masks, List	LSAM									
Databases, List	LSDB									

(3 of 7)

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Modify Transaction Logging	MDLG								buffered-log flags ²	
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
ontape	ONTP									command line
onunload	ONUL									command line
Database, Open	OPDB	dbname							exclusive-flag	dbpassword
Row, Read	RDRW	dbname	tabid		partno	frag_id	rowid ¹⁴			
Optical Cluster, Release	RLOP	family name					volume number			
Transaction, Rollback	RLTX									

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 ⁵					revokees ⁴
Revoke Fragment Access	RVFR	dbname	tabid	fragment	privilege ^{5, 14}			revoker		revokees ^{4, 14}
Revoke Role	RVRL	dbname		rolename				revoker		revokees ⁴
Revoke Table Access	RVTB	dbname	tabid		privilege ^{5, 14}			revoker	drop-flags ²¹	revokees ^{4, 14}
Stored Procedure, System Command	SCSP									command string
Constraint, Set	STCN	dbname							constraint mode ¹¹	constraint names
Set Debug File	STDF	dbname								file path
Set Database Password	STDP	dbname						user id		
Set Dataskip	STDS								skip flags ¹⁶	dbspacelist
Set Explain	STEX								explain flags ¹²	
Isolation Level, Set	STIL				isolation level ³					

(5 of 7)

Event	Mnemonic	dbname	tabid	objname	extra_1	partno	row_num	login	flags	extra_2
Set Lock Mode	STLM								wait flags ¹³	
Set Object Mode	STOM	dbname	tabid		command mode flag ²³				object typeflag ²⁴	object names
Stop Statement	STOP	dbname	tabid							
Set Pdqpriority	STPR								prlevel ¹⁷	
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 ²⁰				mode flags ¹⁹	
Optical Cluster, Time	TMOP								time flag ¹³	
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 ¹⁴		new rowid	new partno

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							

(7 of 7)

NOTES

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

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

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

⁴ Grantees, Revokees, Select Columns, Update Columns: These can be lists of comma-separated names. If longer than 166 characters, the audit processing described in "[Audit Analysis with SQL](#)" on page 3-6 truncates the lists to 166 characters.

⁵ Database Privileges: Table-Level Privileges:
1 Select
2 Insert
4 Delete
8 Update
16 Alter
32 Index
64 Reference
4096 Execute Procedure (When Grant privilege is executed. Table id refers to the procedure ID.)

Database-Level Privileges:
256 Connect
512 DBA
1024 Resource

⁶ Log Status: **1** Logging on
2 Buffered logging
4 ANSI-compliant

⁷ Synonym Type: **0** Private
1 Public

⁸ Lock Mode: **0** Exclusive
1 Shared

⁹ Cluster Flag: **0** Not cluster
1 Cluster

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

- 11 Constraint Mode:
 0 Deferred
 1 Immediate
- 12 Explain Flag:
 0 Explain turned off
 1 Explain turned on
- 13 Wait Flag:
 -1 Wait forever
 0 Do not wait
 >0 Waiting period (in seconds)
- 14 If the user request is turned down because of the authorization, those fields are either 0 or blank, depending on the data type.
- 15 Fragmentation Flag:
 0 Not fragmented
 1 In dbspace
 2 Fragment by round robin
 4 Fragment by expression
 8 Fragment same as table
- 16 Skip Flag:
 0 DATASKIP for all the dbspaces is turned OFF
 1 DATASKIP for the following dbspaces is turned ON
 2 DATASKIP for all the dbspaces is turned ON
 3 DATASKIP is set to the default
- 17 Priority Level:
 -1 PDQPRIORITY is set to the default
 0 PDQPRIORITY is turned OFF
 1 PDQPRIORITY is LOW
 100 PDQPRIORITY is HIGH
 n any other positive integer less than 100 that the user entered in the SET PDQPRIORITY statement
- 18 Operation Type:
 4 Add a new fragment
 8 Modify fragmentation
 16 Drop a fragment
 32 Initialize fragmentation
 64 Attach table(s)
 128 Detach fragment
- 19 Mode Flag:
 0 Read Write if operation is Set Access Mode: Dirty Read if operation is Set Isolation Level
 1 Read Only if operation is Set Access Mode; Committed Read if operation is Set Isolation Level
 2 Cursor Stability
 3 Repeatable Read
- 20 Operation:
 0 Set Access Mode
 1 Set Isolation Level
- 21 Dropflags:
 0 Cascade
 1 Restrict
- 22 Command Mode Flag
 1 Disabled
 2 Filtering without error
 4 Filtering with error
 8 Enabled
- 23 Object Type Flag
 1 Constraint
 2 Index
 3 Constraints and indexes
 4 Trigger
 5 Triggers and constraints
 6 Triggers and indexes
 7 All

The ADTCFG File



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

Tip: In earlier versions of this product, the auditing configuration parameters discussed in this appendix were in the ONCONFIG configuration file.

ADTCFG Parameters

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

- | | |
|------------------------|--|
| <i>default value</i> | is the default value that appears in the adtcfg.std file. |
| <i>if not present</i> | is the value that is supplied if the parameter is missing from your ADTCFG file. |
| <i>units</i> | is the units in which the parameter is expressed. |
| <i>separators</i> | is the separator(s) that can be used when the parameter value has several parts. Do <i>not</i> use white space within a parameter value. |
| <i>range of values</i> | is the valid values for this parameter. |

<i>takes effect</i>	is the time at which a change to the value of the parameter actually affects the operation of the database server.
<i>utility</i>	is the command-line utility that you can use to change the value of the parameter.
<i>refer to</i>	is a cross-reference to further discussion.

ADTCFG File Conventions

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

```
PARAMETER_NAME parameter_value# comment
```

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

For information about additional Dynamic Server configuration parameters, see your [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, the database server writes a message of the failure into the message log. It continues to process the thread.

1 = *halt* error mode: suspend thread processing

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

3 = *halt* error mode: shut down system

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

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

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

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

ADTMODE

<i>default value</i>	0
<i>range of values</i>	0 through 8
	0 = auditing disabled
	1 = database-server-managed auditing on; starts auditing for all sessions
	2 = operating-system-managed auditing on (UNIX systems only)
	3 = database-server-managed auditing on; audits DBSSO actions
	4 = operating-system-managed auditing on; audits DBSSO actions (UNIX systems only)
	5 = database-server-managed auditing on; audits database server administrator actions
	6 = operating-system-managed auditing on; audits database server administrator actions (UNIX systems only)
	7 = database-server-managed auditing on; audits DBSSO and database server administrator actions
	8 = operating-system-managed auditing on; audits DBSSO and database server administrator actions (UNIX systems only)
<i>takes effect</i>	When onaudit is run to change the value, or after shared memory is initialized
<i>utility</i>	onaudit (<code>onaudit -l <i>auditmode</i></code>)

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

UNIX

ADTPATH (UNIX Only)

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 in which the database server can save audit files.

Database-server-managed auditing must be on for changing the ADTPATH value with **onaudit**.

UNIX

ADTSIZE (UNIX Only)

default value 10,240

units Bytes

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

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

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

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

Index

A

AAO

See Audit analysis officer.

aaodir directory 2-5

Access privileges, Windows

NT 1-30, 2-4, 2-5

Access to audit data,

controlling 1-23, 1-25, 3-9

Access to audit records,

controlling 1-23

Adding audit masks 2-14

Administrative roles

audit analysis officer 2-5

database administrator 2-6

database server administrator 2-4

database system security officer

listed 1-9

operating-system

administrator 2-6

Administrator

audit analysis officer 2-5

database 2-6

database server 2-4

database system security

officer 2-4

operating system 2-6

Administrator group 1-21

ADTCFG file

aaodir directory 2-5

ADTERR parameter 2-20, 2-21

ADTMODE parameter 2-20, 2-21

ADTPATH parameter 2-20

ADTSIZE parameter 2-20

audit configuration

UNIX 1-24, 4-15

Windows NT 1-24, 4-15

conventions used B-1, B-2

description of B-2

UNIX audit file size 1-20

white space B-2

ADTERR configuration

parameter 2-20, 2-21, B-3

ADTMODE configuration

parameter 2-20, 2-21, B-4

ADTPATH configuration

parameter 2-20, B-5

ADTSIZE configuration

parameter 2-20, B-5

Aggregation 1-31

ANSI compliance level Intro-14

Audit

audit trail administration 2-17

configuration, displaying 2-20

configuration, overview 1-17

configuring auditing 1-17

features 1-3

minimum events to audit 1-14

performance 1-13

process for 1-7

reasons for 1-3

record format 3-3

starting a new UNIX audit log

file 2-21

turning on auditing 2-13

Audit analysis

creating a data file 3-6

importance of 1-27

loading audit data into a

database 3-10

overview 1-27

preparing for 1-28

records indicating event

failure 1-30

- records indicating event
 - success 1-30
- strategies for 1-30
- with SQL
 - creating a command file 3-9
 - creating a database and table 3-7
 - description 3-6
 - performing 3-10
 - preparing for 3-6
 - without database 3-5
 - without SQL 3-5
- Audit analysis officer
 - role description 2-5
 - security threats 1-34
 - UNIX 2-5
 - Windows NT 2-5
- Audit configuration
 - changing from a command line 4-14
 - maintaining 2-19
 - showing
 - from a command line 2-20, 4-13
 - with onshowaudit 4-13
 - UNIX onaudit output 4-13
 - Windows NT onaudit output 4-14
- Audit data
 - controlling access to 3-9
 - creating a table for 3-7
 - loading into database 3-10
 - privileges to protect 3-9
- Audit error mode
 - and onaudit 4-14
 - changing 2-23
 - in ADTCFG file B-3
 - setting 2-11
- Audit events
 - alphabetical listing of codes A-1
 - fields shown A-6
 - listed A-6
 - minimum ones to audit 1-14
- Audit files, UNIX
 - audit trail 1-7, 1-22
 - controlling access to 1-25
 - directory
 - specifying with ADTPATH B-5
 - specifying with onaudit 4-15
 - error modes when writing to 1-23
 - extracting information with onshowaudit 4-17
 - location of 1-20
 - naming 1-21
 - properties of 1-19
 - specifying maximum size
 - with ADTSIZE B-5
 - with onaudit 4-15
 - starting
 - new file 1-20
 - with onaudit 4-12
 - storage
 - in database server 4-12
 - in operating system 4-12
 - write errors 4-16
- Audit instructions
 - minimum events to audit 1-14
 - resource and performance implications 1-13
 - who sets 1-12
- Audit level, setting 2-12
- Audit masks
 - adding 2-14
 - compulsory masks 1-11
 - conflict in audit instructions 1-11, A-1
 - creating a template 2-15
 - creating a user mask from a template mask 2-15
 - creating from a command line 4-6
 - deleting 2-19, 4-11
 - displaying 2-17
 - how to use 1-16
 - individual user mask 1-10
 - maintaining 2-14
 - modifying
 - command syntax for 4-10
 - from a command line 2-18
 - from an input file 2-16
 - how to 2-18
 - restricted names 1-12
 - setting up default and compulsory 2-10
 - showing 4-5
 - specification with onaudit 4-7
 - templates 1-11
 - types, listed 1-10
 - user informix 1-11
 - _default mask 1-10
 - _exclude mask 1-10
 - _require mask 1-10
- Audit mode, changing
 - UNIX 2-22
 - Windows NT 2-22
- Audit records, controlling access to 1-23, 1-25
- Audit trail
 - administration 2-14
 - controlling access to 1-25
 - event log, Windows NT 1-7
 - extracting information with onshowaudit 4-18
 - operating-system, UNIX 1-7
 - reviewing 1-8
 - starting a new UNIX file 2-21
 - starting auditing from a command line 4-12
 - storing
 - in database server 1-18
 - in operating system 1-18, 1-19
 - UNIX file permissions 1-23, 1-25, 1-26
 - UNIX files 1-22
 - Windows NT access privileges 1-26
 - Windows NT event log 1-22
- Auditing
 - ADTCFG file
 - UNIX 1-24, 4-15
 - Windows NT 1-24, 4-15
 - creating user masks from template masks 2-15
 - displaying fragmentation information 1-15
 - error mode levels 4-14
 - granularity 1-15
 - operating system versus database server 1-18
 - setting the level 2-12
 - setting up 2-10
 - specifying directory
 - with ADTPATH B-5
 - specifying UNIX directory
 - with onaudit 4-15
 - turning off 2-23, 4-16
 - turning on 2-13, 4-16
 - turning on or off 1-17

B

Browsing 1-31

C

Changing the audit error mode 2-23
Changing the audit mode
 UNIX 2-22
 Windows NT 2-22
Changing the system audit
 configuration 4-14
Command files
 creating for dbload 3-9
 use with dbload 3-9
Command-line conventions
 elements of Intro-9
 example diagram Intro-10
 how to read Intro-10
Comment icons Intro-7
Compliance with industry
 standards Intro-14
Compulsory audit masks
 setting up 2-10
 when applied 1-10
Configuration parameter
 ADTERR B-3
 ADTMODE B-4
 ADTPATH B-5
 ADTSIZE B-5
 listed 2-20, 2-21
Configuring role separation 2-8
Controlling access to audit files 1-23
Controlling access to audit
 records 1-23
Controlling access to the audit
 trail 1-25
Creating a data file 3-6
Creating a database and table for
 audit data 3-7
Creating a user mask from a
 template mask 2-15
Creating an audit mask from a
 command line 4-6

D

DAC
 See Discretionary Access Control.
Data
 audit, loading into database 3-10
 creating a file for dbload 3-6
 extracting with onshowaudit 3-5
Database
 creating for audit records 3-7
 demonstration Intro-5
 sysmaster 2-20
Database administrator
Database server
 administrator 2-4
 audit log 4-19
 auditing 1-18, 4-17
 managing auditing
 with ADTMODE B-4
 with onaudit 4-16
 monitoring events and users 2-12
Database server administrator
 security threats 1-34
Database system security officer
 role description 2-4
 security threats 1-34
 UNIX 2-4
 Windows NT 2-4
DBA
 See Database administrator.
dbload utility
 creating a command file for 3-9
 creating a data file for 3-6
 creating a database and table
 for 3-7
 creating onshowaudit output files
 for 4-17
 format of data file for fragmented
 tables 3-7
 loading audit data into a
 database 3-10
 redirecting onshowaudit
 output 4-17
DBMS security threats 1-33
DBSSO
 See Database system security
 officer.

Default audit mask 1-10
 setting up 2-10
 when applied 1-10
Default locale Intro-4
Deleting audit masks 2-19, 4-11
Demonstration database Intro-5
Directory
 aadir 2-5
 specifying for UNIX audit
 files 2-11
 specifying with onaudit 4-15
Directory, UNIX
 specifying with ADTPATH B-5
Discretionary Access Control 1-33
Displaying
 audit configuration 2-20, 4-13
 audit masks 2-17, 4-5
Distributed databases
 configuration security
 threats 1-37
Documentation
 conventions
 command-line Intro-8
 icon Intro-7
 typographical Intro-6
 on-line notes Intro-13
 related reading Intro-14
 types of
 printed manuals Intro-12
Documentation notes Intro-13
Documentation notes, program
 item Intro-13
Documentation, types of
 documentation notes Intro-13
 error message files Intro-12
 machine notes Intro-13
 on-line manuals Intro-11
 release notes Intro-13

E

Enable Role Separation check
 box 1-9, 2-9
Enforcing role separation 2-8
Environment variable
 INF_ROLE_SEP 1-9, 2-8
 NODEFDAC 3-9
en_us.8859-1 locale Intro-4

Error messages
files Intro-12
Error messages log, size of 1-23
Error mode
and ADTERR B-3
and onaudit 4-14
changing 2-23
continue 1-24
halt 1-24
implications of 2-11
setting 2-11
when writing to a UNIX audit
file 1-23
when writing to the Windows NT
event log 1-23
Event codes, alphabetical
listing A-1
Event failure 1-30
Event log, Windows NT 1-7, 1-21,
1-22, 1-23, 1-26, 4-16
Event success 1-30
Event Viewer administrative
tool 1-30
Events
fields shown A-6
level of auditing for specified 1-16
mnemonics listed A-1
which ones to audit 1-14
Exclude audit mask 1-10

F

Feature icons Intro-8
Fields for audit events A-6
File
ADTCFG 1-20
audit trail, starting a new UNIX
file 2-21
data, creating for dbload 3-6
input
for modifying masks 2-16
for onaudit 4-9
UNIX audit
controlling access to 1-23, 1-25
error modes when writing
to 1-23
location of 1-20

naming 1-21
starting new file 1-20
starting with onaudit 4-12
storage in database server 4-12
storage in operating system 4-12
FILE statement 3-10
finderr script Intro-12
Format
for audit records 3-3
for dbload data file 3-7
for onaudit input file 4-9
Fragmentation, information in
audit events 1-15

G

Global Language Support
(GLS) Intro-4
Group informix, UNIX database
server administrator 2-4
Guidelines for assigning roles 2-7

H

Halt modes 4-16

I

Icons
comment Intro-7
feature Intro-8
platform Intro-8
product Intro-8
Industry standards, compliance
with Intro-14
Informix Find Error utility Intro-12
informix user account 1-22
INFORMIXDIR/bin
directory Intro-5
INF_ROLE_SEP environment
variable 1-9, 2-8
Input file for onaudit utility 4-9
Insider attack 1-31
ISO 8859-1 code set Intro-4

L

Level of auditing, determining 2-12
Loading onshowaudit data into a
database table 3-10
Locale Intro-4
LocalSystem user account 1-21, 1-22

M

Machine notes Intro-13
Malicious software security
threats 1-35
Manual
purpose of Intro-3
types of users Intro-3
Mask
creating
template 2-15
user mask from a template
mask 2-15
user mask without a template
mask 2-16
with onaudit 4-6
deleting 2-19, 4-11
displaying 2-17
for user informix 1-11
how to use 1-16
modifying
from an input file 2-16
from the command line 2-18
with onaudit 4-10
onaudit input-file format 4-9
setting up compulsory 2-10
setting up default 2-10
showing with onaudit 4-5
specification with onaudit 4-7
template 1-11
types, listed 1-10
user 1-10
_default 1-10
_exclude 1-10
_require 1-10
Message log 4-16
Message Server service 1-22
Mnemonics, alphabetical
listing A-1
Modifying audit masks 2-18, 4-10

N

Named pipes interprocess communications 1-22
New features, Version 7.3 Intro-5
NODEFDAC environment variable 3-9

O

Obsolete user security threats 1-36
onaudit utility
 adding audit events to audit masks 2-10
 ADTERR parameter B-3
 ADTMODE parameter B-4
 ADTPATH parameter B-5
 ADTSIZE parameter B-5
 auditing mode levels 4-15
 changing the audit error mode 2-23
 changing the audit mode 2-22
 changing the system audit configuration 4-14
 creating a template mask 2-15
 creating a user mask from a template mask 2-15
 creating a user mask without a template mask 2-16
 creating an audit mask 4-6
 deleting audit masks 2-19, 4-11
 description of 4-3, 4-4
 displaying audit masks 2-17
 displaying the audit configuration 2-20
 error-mode levels 4-14
 fragmentation information 1-15
 input-file format 4-9
 level of auditing for certain events 1-16
 modifying masks 2-16, 4-10
 railroad diagram of 4-4
 setting the error mode 2-11
 showing audit masks from a command line 4-5
 showing the auditing configuration 4-13

 specifying a directory for UNIX audit files 2-11
 starting a new UNIX audit file 4-12
 starting a new UNIX audit-trail file 2-21
 storage of audit records 4-12
 syntax 4-4
 turning off auditing 2-23
 turning on auditing 2-13
 UNIX operations 4-4
 used by AAO 2-5
 used by DBSSO 2-4
 who can run 4-5
 Windows NT operations 4-4
On-line manuals Intro-11
onshowaudit utility
 description of 4-3
 extracting data for audit analysis 3-5
 format of data file for fragmented tables 3-7
 listing of audit events for analysis A-6
 syntax 4-18
 used by AAO 2-5
 using dbload with 3-7
 who can run 4-18
Operating system
 audit log 4-19
 audit record format 3-3
 auditing 1-18
 coordinating auditing between AAO and OSA 2-5
 managing auditing with ADTMODE B-4 with onaudit 4-16
 protected subsystem for audit trail 1-29
 storing audit records 1-19, 2-11
Operating-system administrator
 role defined 2-6
 security threats 1-34
Operating-system audit trail, UNIX 1-7
OSA
 See Operating-system administrator.

P

Parameters, configuration
 ADTERR B-3
 ADTMODE B-4
 ADTPATH B-5
 ADTSIZE B-5
 described B-1
 listed 2-20, 2-21
 Path, specifying for auditing with ADTPATH B-5 with onaudit 4-15
 Performance implications of auditing 1-13
 Performing SQL audit analysis 3-10
 Permissions
 UNIX 2-5
 Permissions, UNIX 1-30, 2-4
 Platform icons Intro-8
 Preparing for audit analysis 1-28, 3-6
 Primary security threats 1-33
 Printed manuals Intro-12
 Privileged activity security threats 1-34
 Privileged environment, security threat from untrusted software 1-37
 Privileges to protect audit data 3-9
 Product icons Intro-8
 Program group
 documentation notes Intro-13
 release notes Intro-13
 Purpose of manual Intro-3

R

Registry settings, Windows NT 2-4, 2-5, 2-9
Release notes Intro-13
Release notes, program item Intro-13
Remote access to data, security threat 1-36
Require audit mask 1-10
Resource implications of auditing 1-13

- Responding to security
 - problems 1-32
- rofferr script Intro-12
- Role separation
 - Windows NT 2-4, 2-9
- Role Separation dialog box 1-9, 2-9
- Roles
 - administrative, listed 1-9
 - assigning 2-7
 - audit analysis officer 2-5
 - configuring and enforcing 2-8
 - database administrator 2-6
 - database server administrator 2-4
 - database system security officer 2-4
 - no separation, security configuration for 1-25
 - operating-system administrator 2-6
 - separation 1-26, 2-7

S

- Security log, Windows NT 1-21, 1-22
- Security subsystem, Windows NT 1-21
- Security threats
 - aggregation 1-31
 - audit analysis officer 1-34
 - browsing 1-31
 - database server administrator 1-34
 - database system security officer 1-34
- DBMS 1-33
- distributed databases
 - configuration 1-37
- granting remote access to data 1-36
- insider attack 1-31
- introduction of malicious software 1-35
- obsolete user 1-36
- OSA 1-34
- primary 1-33
- privileged activity 1-34

- responses to 1-32
- setting the auditing level 2-12
- shared-memory connection 1-35
- untrusted software in privileged environment 1-37

Shared-memory connection 1-35

Showing

- audit masks 4-5
- auditing configuration 4-13

Size, specifying maximum for

- UNIX audit files
 - with ADTSIZE B-5
 - with onaudit 4-15

SMI sysadinfo table 2-20

Software dependencies Intro-4

Specification, audit mask 4-7

SQL statement

- CREATE DATABASE 3-7
- GRANT 3-9
- REVOKE 3-9

Starting a new audit trail 2-21

Starting a new UNIX audit file 4-12

Storage of UNIX audit files

- in database server 4-12
- in operating system 4-12
- new file (-n) option 4-12

stores7 database Intro-5

Strategies for audit analysis 1-30

Syntax

- onaudit utility 4-4
- onshowaudit utility 4-17

sysadinfo table 2-20

sysmaster database, sysadinfo table 2-20

System log, Windows NT 1-22

T

- Table
 - creating for audit data 3-7
 - sysadinfo 2-20
- Template audit masks 1-11
 - creating from user masks 2-15
 - creating with onaudit 2-15
 - description 1-11

U

- UNIX operating system
 - directory for on-line files Intro-13
 - reading error messages Intro-12
- User informix
 - making a mask for 1-11
 - retrieving audit configuration information 2-20
 - running onaudit 4-5
 - running onshowaudit 4-18
- User informix, Windows NT database server administrator 2-4
- User mask
 - creating from a template mask 2-15
 - creating without a template mask 2-16
- Users
 - auditing 1-10, 4-17
 - privileged 2-7
 - system 2-6
- Utility
 - finderr Intro-12
 - Informix Find Error Intro-12
 - rofferr Intro-12

W

- White space in ADTCFG file B-2
- Windows NT
 - program groups for on-line notes Intro-13
 - reading error messages Intro-12
- Windows NT services 1-21

X

- X/Open compliance level Intro-14