Informix Product Family
Informix
Version 11.70

IBM Informix Security Guide

SC27-3562-04

Informix Product Family
Informix
Version 11.70

IBM Informix Security Guide

SC27-3562-04

Note
Before using this information and the product it supports, read the information in "Notices" on page B-1.

This edition replaces SC27-3562-03.

Contents

Part 2. Auditing data security

Introduction

About this publication

This publication documents methods for keeping your data secure by preventing unauthorized viewing and altering of data or database objects.

This publication also documents the secure-auditing facility of the database server.

This manual is not a computer-security or trusted-facility-administration training manual.

Types of users

This publication is written for the following users:

- Database administrators
- Database server administrators
- Operating system administrators
- Database system security officers

This manual is written with the assumption 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
- Some experience with computer programming
- Some experience with database server administration, operating-system administration, or network administration

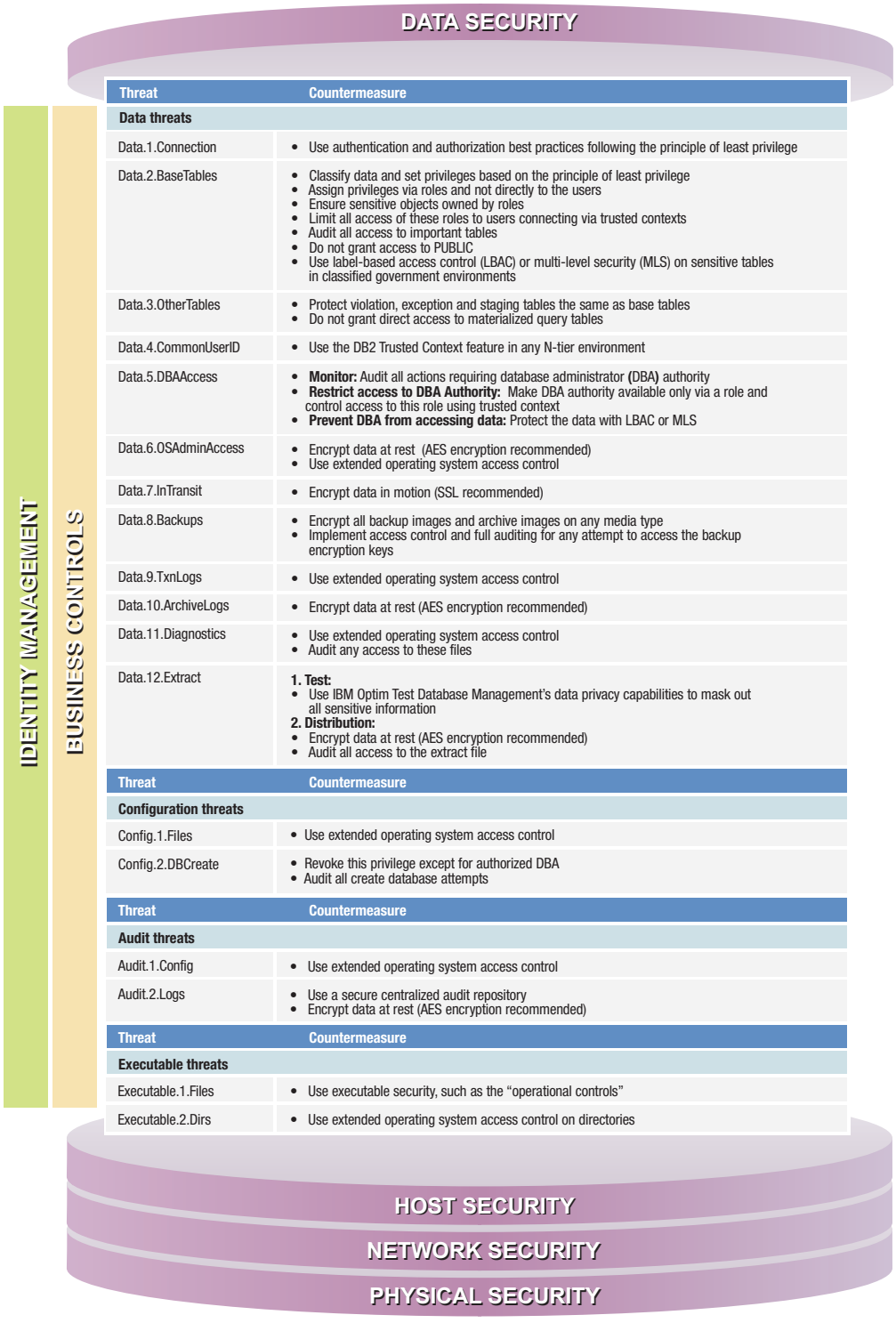
If you have limited experience with relational databases, SQL, or your operating system, see the online information center for a list of supplementary titles.

Prerequisites for a secure database server

You must ensure that the overall database environment is properly secured before implementing many of the features detailed in IBM® Informix® security documentation.

Database technology comprises interdependent components. Security must exist on each of these components at each layer in the environment to make a truly secure system. Security measures should include network, host, and physical security; identity management; and business controls.

For information about some steps you can take to help secure your database server and the data on your system, see the IBM Data Server Security Blueprint. Download the complete and most up-to-date blueprint at ftp://ftp.software.ibm.com/software/data/db2imstools/whitepapers/data-server-security-blueprint-paper.pdf. A list of recommended precautions is reproduced in Figure 1 on page x.



Threat	Countermeasure
Data threats	
Data.1.Connection	• Use authentication and authorization best practices following the principle of least privilege
Data.2.BaseTables	• Classify data and set privileges based on the principle of least privilege • Assign privileges via roles and not directly to the users • Ensure sensitive objects owned by roles • Limit all access of these roles to users connecting via trusted contexts • Audit all access to important tables • Do not grant access to PUBLIC • Use label-based access control (LBAC) or multi-level security (MLS) on sensitive tables in classified government environments
Data.3.OtherTables	• Protect violation, exception and staging tables the same as base tables • Do not grant direct access to materialized query tables
Data.4.CommonUserID	• Use the DB2 Trusted Context feature in any N-tier environment
Data.5.DBAAccess	• **Monitor:** Audit all actions requiring database administrator **(DBA)** authority • **Restrict access to DBA Authority:** Make DBA authority available only via a role and control access to this role using trusted context • **Prevent DBA from accessing data:** Protect the data with LBAC or MLS
Data.6.OSAdminAccess	• Encrypt data at rest (AES encryption recommended) • Use extended operating system access control
Data.7.InTransit	• Encrypt data in motion (SSL recommended)
Data.8.Backups	• Encrypt all backup images and archive images on any media type • Implement access control and full auditing for any attempt to access the backup encryption keys
Data.9.TxnLogs	• Use extended operating system access control
Data.10.ArchiveLogs	• Encrypt data at rest (AES encryption recommended)
Data.11.Diagnostics	• Use extended operating system access control • Audit any access to these files
Data.12.Extract	**1. Test:** • Use IBM Optim Test Database Management's data privacy capabilities to mask out all sensitive information **2. Distribution:** • Encrypt data at rest (AES encryption recommended) • Audit all access to the extract file
Threat	**Countermeasure**
Configuration threats	
Config.1.Files	• Use extended operating system access control
Config.2.DBCreate	• Revoke this privilege except for authorized DBA • Audit all create database attempts
Threat	**Countermeasure**
Audit threats	
Audit.1.Config	• Use extended operating system access control
Audit.2.Logs	• Use a secure centralized audit repository • Encrypt data at rest (AES encryption recommended)
Threat	**Countermeasure**
Executable threats	
Executable.1.Files	• Use executable security, such as the "operational controls"
Executable.2.Dirs	• Use extended operating system access control on directories

Figure 1. Data security threats and countermeasures

What's new for security in Informix, Version 11.70

This publication includes information about new features and changes in existing functionality.

The following changes and enhancements are relevant to this publication. For a complete list of what's new in this release, see the release notes or the information center at http://publib.boulder.ibm.com/infocenter/idshelp/v117/topic/com.ibm.po.doc/new_features.htm.

Table 1. What's new in IBM Informix Security Guide for Version 11.70.xC4

Overview	Reference
Global Security Kit (GSKit) support Informix 11.70 now supports GSKit version 7 or later, and ships with GSKit version 8. You can set the GSKIT_VERSION configuration parameter so that the database server uses a version other than GSKit version 8.	See "IBM Global Security Kit" on page 2-14

Table 2. What's new in IBM Informix Security Guide for Version 11.70.xC3

Overview	Reference
Restrict operating system properties for mapped users (UNIX and Linux) The operating system administrator can now use the `/etc/informix/allowed.surrogates` file to control which operating system users and groups can act as surrogates for mapped users. The improved control makes root installations of Informix more secure by preventing the database security administrator from specifying surrogates that could compromise operating system security.	See "Mapped user surrogates in the allowed.surrogates file (UNIX, Linux)" on page 4-5
Retaining numbers for audit log files Audit log file numbers are no longer reused if the database server restarts after audit log files are removed. The `adtlog.server` file in the `$INFORMIXDIR/aaodir` directory maintains the number of the most recent audit log file. The `adtlog.server` file saves system resources whenever the database server restarts because the database server no longer checks each audit log file number before assigning a number to the new log file.	See "Properties of audit files" on page 7-11

Table 3. What's new in IBM Informix Security Guide for Version 11.70.xC2

Overview	Reference
Fewer users require administrator access (UNIX, Linux) Informix software can be installed and administered without root privileges on UNIX and Linux operating systems. The user who performs the non-root installation is not required to be a system administrator, and this user is the database server administrator (DBSA) so that the database server can run on the computer without user **informix** and group **informix**. This feature is useful for deploying the server in embedded software solutions or for situations where running software that uses root privileges raises security concerns. A non-root installation does not support high-availability replication, the OpenAdmin Tool (OAT) for Informix, the **ON-Bar** utility, and role separation.	See "Connections to a non-root installation (UNIX, Linux)" on page 4-8.

Table 4. What's new in IBM Informix Security Guide for Version 11.70.xC1

Overview	Reference
Simplified administration of users without operating system accounts (UNIX, Linux) In previous releases, each user who needed to access the database server also needed an operating system account on the host computer. Now you can configure Informix so that users who are authenticated by an external authentication service (such as Kerberos or Microsoft Active Directory) can connect to Informix. The new USERMAPPING configuration parameter specifies whether or not such users can access the database server, and whether any of those users can have administrative privileges. When Informix is configured to allow user mapping, you can still control which externally authenticated users are allowed to connect to Informix and their privileges.	See "Mapped users (UNIX, Linux)" on page 4-4.
Selective row-level auditing The database system security officer (DBSSO) can configure auditing so that row-level events are recorded for designated tables, rather than for all tables used by the database server. By selecting only the tables that you want to audit on the row level, you can improve database server performance, simplify audit trail records, and mine audit data more effectively. Previously, there was no way to enable auditing so that it excluded audit events on tables that you did not want to monitor with the **onaudit** utility.	See "Selective row-level auditing" on page 7-3.
Trusted connections improve security for multiple-tier application environments You can define trusted contexts, which can then be used to establish trusted connections between an application server and the Informix database server on a network. Trusted connections let you set the identity of each specific user accessing a database through the middle-tier server, which facilitates discretionary access control and auditing based on user identity. Without a trusted connection in such an environment, each action on a database is performed with the single user ID of the middle-tier server, potentially lessening granular control and oversight of database security.	See "Trusted-context objects and trusted connections" on page 4-14
New editions and product names IBM Informix Dynamic Server editions were withdrawn and new Informix editions are available. Some products were also renamed. The publications in the Informix library pertain to the following products: • IBM Informix database server, formerly known as IBM Informix Dynamic Server (IDS) • IBM OpenAdmin Tool (OAT) for Informix, formerly known as OpenAdmin Tool for Informix Dynamic Server (IDS) • IBM Informix SQL Warehousing Tool, formerly known as Informix Warehouse Feature	For more information about the Informix product family, go to http://www.ibm.com/software/data/informix/.

Example code conventions

Examples of SQL code occur throughout this publication. Except as noted, the code is not specific to any single IBM Informix application development tool.

If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores_demo
...

DELETE FROM customer
   WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

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

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

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the documentation for your product.

Additional documentation

Documentation about this release of IBM Informix products is available in various formats.

You can access or install the product documentation from the Quick Start CD that is shipped with Informix products. To get the most current information, see the Informix information centers at ibm.com®. You can access the information centers and other Informix technical information such as technotes, white papers, and IBM Redbooks® publications online at http://www.ibm.com/software/data/sw-library/.

Compliance with industry standards

IBM Informix products are compliant with various standards.

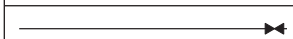
IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of IBM Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment (CAE) standards.

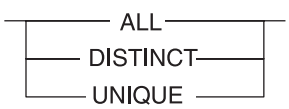
The IBM Informix Geodetic DataBlade® Module supports a subset of the data types from the *Spatial Data Transfer Standard (SDTS)—Federal Information Processing Standard 173*, as referenced by the document *Content Standard for Geospatial Metadata*, Federal Geographic Data Committee, June 8, 1994 (FGDC Metadata Standard).

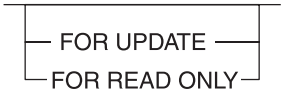
Syntax diagrams

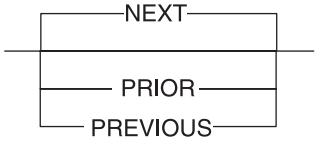
Syntax diagrams use special components to describe the syntax for statements and commands.

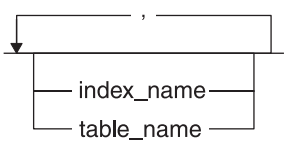
Table 5. Syntax Diagram Components

Component represented in PDF	Component represented in HTML	Meaning
	`>>----------`	Statement begins.
	`---------->`	Statement continues on next line.
	`>----------`	Statement continues from previous line.
	`----------><`	Statement ends.
	`--------SELECT----------`	Required item.
	`--+-----------------+---` `  '------LOCAL------'`	Optional item.

	`---+-----ALL-------+---`
	`   +--DISTINCT-----+`
	`   '---UNIQUE------'`
		Required item with choice. Only one item must be present.

	`---+------------------+---`
	`    +--FOR UPDATE-----+`
	`    '--FOR READ ONLY--'`
		Optional items with choice are shown below the main line, one of which you might specify.

	`    .---NEXT---------.`
	`----+----------------+---`
	`    +---PRIOR--------+`
	`    '---PREVIOUS-----'`
		The values below the main line are optional, one of which you might specify. If you do not specify an item, the value above the line is used by default.

	`  .-------,-----------.`
	`  V                  |`
	`---+-----------------+---`
	`    +---index_name---+`
	`    '---table_name---'`
		Optional items. Several items are allowed; a comma must precede each repetition.

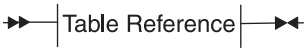
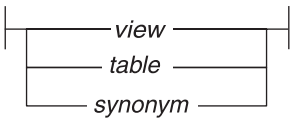
	`>>-| Table Reference |-><`	Reference to a syntax segment.

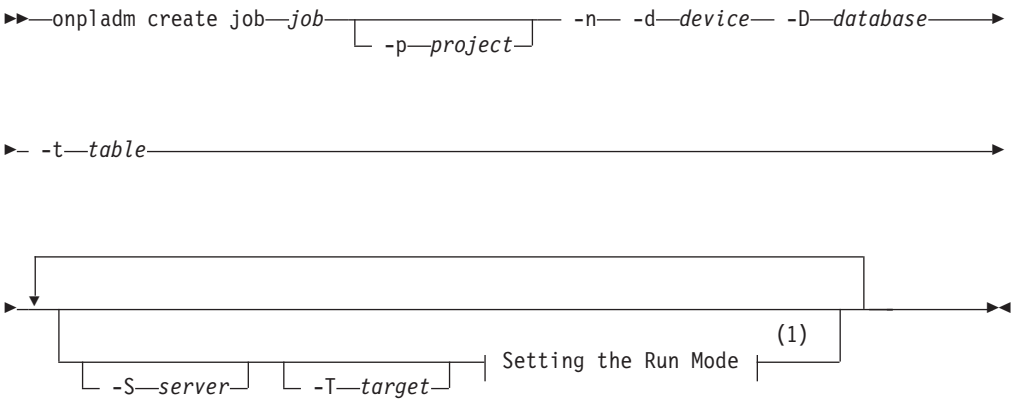
Table Reference
 
	`Table Reference`
	`|--+-----view--------+--|`
	`   +------table------+`
	`   '----synonym------'`
		Syntax segment.

How to read a command-line syntax diagram

Command-line syntax diagrams use similar elements to those of other syntax diagrams.

Some of the elements are listed in the table in Syntax Diagrams.

Creating a no-conversion job

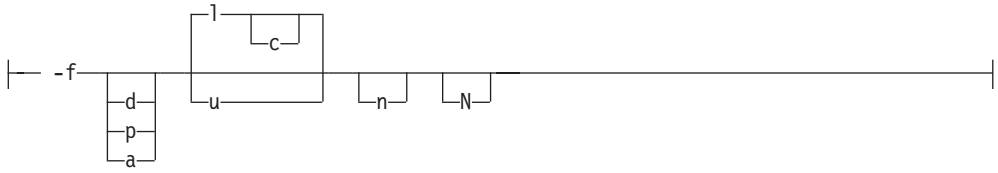


Notes:

1 See page Z-1

This diagram has a segment named “Setting the Run Mode,” which according to the diagram footnote is on page Z-1. If this was an actual cross-reference, you would find this segment on the first page of Appendix Z. Instead, this segment is shown in the following segment diagram. Notice that the diagram uses segment start and end components.

Setting the run mode:



To see how to construct a command correctly, start at the upper left of the main diagram. Follow the diagram to the right, including the elements that you want. The elements in this diagram are case-sensitive because they illustrate utility syntax. Other types of syntax, such as SQL, are not case-sensitive.

The Creating a No-Conversion Job diagram illustrates the following steps:

1. Type **onpladm create job** and then the name of the job.
2. Optionally, type **-p** and then the name of the project.
3. Type the following required elements:
   - **-n**
   - **-d** and the name of the device
   - **-D** and the name of the database
   - **-t** and the name of the table

4. Optionally, you can choose one or more of the following elements and repeat them an arbitrary number of times:
   - **-S** and the server name
   - **-T** and the target server name
   - The run mode. To set the run mode, follow the Setting the Run Mode segment diagram to type **-f**, optionally type **d**, **p**, or **a**, and then optionally type **l** or **u**.
5. Follow the diagram to the terminator.

Keywords and punctuation

Keywords are words reserved for statements and all commands except system-level commands.

When a keyword appears in a syntax diagram, it is shown in uppercase letters. When you use a keyword in a command, you can write it in uppercase or lowercase letters, but you must spell the keyword exactly as it appears in the syntax diagram.

You must also use any punctuation in your statements and commands exactly as shown in the syntax diagrams.

Identifiers and names

Variables serve as placeholders for identifiers and names in the syntax diagrams and examples.

You can replace a variable with an arbitrary name, identifier, or literal, depending on the context. Variables are also used to represent complex syntax elements that are expanded in additional syntax diagrams. When a variable appears in a syntax diagram, an example, or text, it is shown in *lowercase italic*.

The following syntax diagram uses variables to illustrate the general form of a simple SELECT statement.

When you write a SELECT statement of this form, you replace the variables *column_name* and *table_name* with the name of a specific column and table.

How to provide documentation feedback

You are encouraged to send your comments about IBM Informix user documentation.

Use one of the following methods:

- Send email to docinf@us.ibm.com.
- In the Informix information center, which is available online at http://www.ibm.com/software/data/sw-library/, open the topic that you want to comment on. Click the feedback link at the bottom of the page, fill out the form, and submit your feedback.

- Add comments to topics directly in the information center and read comments that were added by other users. Share information about the product documentation, participate in discussions with other users, rate topics, and more!

Feedback from all methods is monitored by the team that maintains the user documentation. The feedback methods are reserved for reporting errors and omissions in the documentation. For immediate help with a technical problem, contact IBM Technical Support at http://www.ibm.com/planetwide/.

We appreciate your suggestions.

Part 1. Securing data

This section contains information about methods for keeping your data secure by preventing unauthorized viewing and altering of data or other database objects.

Chapter 1. IBM Informix directory security

IBM Informix utilities and product directories are secure by default.

- The database server utilities check security before and after the database server starts. See "Utilities for checking directory security (UNIX and Linux)."
- The directory permissions of the installation path and key subdirectories must meet security requirements to prevent attacks on Informix programs. See "Installation path security requirements (UNIX and Linux)" on page 1-2.
- The **onsecurity** utility checks the security of the directories of the installation path. When you run this utility manually or when you install Informix Version 11.50.xC4 (or later version), you are notified of potentially dangerous directory permissions and how to correct the problems. See "The onsecurity utility (UNIX and Linux)" on page 1-4.
- You cannot continue to use many programs with the database server if a security problem in `$INFORMIXDIR` or its subdirectories arises. See "Securing a nonsecure $INFORMIXDIR and its subdirectories (UNIX and Linux)" on page 1-8
- Most IBM Informix utilities run as secure users and belong to a secure group. See "Users and group membership for running utilities" on page 1-10
- The chunk files that hold the data for Informix must be secure also. See "Security of the chunk files" on page 1-11.
- Use the **DB_LIBRARY_PATH** configuration parameter to control the location from which shared objects, such as external modules, can be loaded. See "Security for loading external modules" on page 1-11.

Utilities for checking directory security (UNIX and Linux)

The database server utilities make security checks before the database server starts.

To provide increased security, key server utilities check if your environment is secure. Before the database server starts, the following settings must be unchanged from the settings established during installation:

- The permissions on directories in the installation path. When you install a new version of your database server, follow the installation instructions to ensure that the permissions of all key files and directories are set appropriately. If you change the path permissions after installation in such a way that the server utilities detect that the path is not secure, Informix will not start.
- The permissions on `$INFORMIXDIR` and its subdirectories. For each directory, the database server checks that the directory exists, that it is owned by user **informix** and the correct group (as shown in "Installation path security requirements (UNIX and Linux)" on page 1-2), and that directory permissions do not include write permissions for the group or other users.
- The permissions on the `onconfig` file.

  The configuration file must belong to the Database Server Administrator (DBSA) group. If the DBSA group is **informix** (the default group), the `onconfig` file must be owned by user **informix**; otherwise, the ownership is not restricted. The file must not have write permissions for others.
- The permissions on the `sqlhosts` file.

Under the default configuration, the sqlhosts file is located in the $INFORMIXDIR/etc directory. The owner must be user **informix**, the group must be either the **informix** group or the DBSA group, and the file must not have public write permissions. If the file is specified through an INFORMIXSQLHOSTS environment variable, the owner and group are not checked; however, public write permissions are not permitted.

- File name lengths.

  The length of the onconfig file name in $INFORMIXDIR/etc must be less than 256 bytes.

If the tests for any of these conditions fail, the utilities exit with an error message.

Utilities check that the path specified by the INFORMIXDIR environment variable is secure whenever you attempt to start major programs like **oninit**, **onmode**, etc. The security check stops programs from starting if the $INFORMIXDIR path is not secure to help prevent the possibility that attackers can change software that is secure to software that is not secure. Use the **onsecurity** utility to diagnose the problem, and in some cases, to change directory permissions.

In rare circumstances, troubleshooting security issues can require that utilities that run as root user or user **informix** can start in a nonsecure environment temporarily (that is, root and user **informix** are not stopped by the utilities that detect a security problem in the $INFORMIXDIR path). See the IFX_NO_SECURITY_CHECK environment variable documentation in the *IBM Informix Guide to SQL: Reference* for more information.

The installation media for Informix Version 11.50.xC4 and later completes a security check on the selected destination path before the binary files are copied to the target host computer. See the security-related documentation in the latest version of *IBM Informix Installation Guide for UNIX, Linux, and Mac OS X* for more information.

The **onsecurity** utility is available on your host computer as a stand-alone tool to check directory permissions of the path specified by the INFORMIXDIR environment variable after you have installed Informix Version 11.50.xC4 and later versions. The **onsecurity** utility is copied to $INFORMIXDIR/bin.

Installation path security requirements (UNIX and Linux)

The owner, group, and write access settings of the directories in the installation path and key subdirectories must be secure to prevent attacks on Informix programs.

Informix checks directory permissions when it is started to help prevent security breaches, such as a denial-of-service attack or a time-of-check, time-of-use (TOCTOU) attack (also known as a race condition).

The installation path is secure when each directory in it (from the root directory to the installation directory) meets all of the following conditions:

- The user that owns the directory is trusted.
- Either the group that owns the directory is trusted or the group cannot write in the directory.
- There is no public write access to the directory. A directory with public write access is inherently not secure because any user can move or rename the directory or a file within it.

The main installation directory must be owned by user **informix**, must belong to group **informix**, and must not have public write permission. Typically, no user requires write permission on the directory, but in many environments user **informix** is granted this permission.

To complete a transaction on the database server that requires trusted privileges, a user must have a user name and belong to a group that matches the names of corresponding, trusted entities that exist on the computer. If a user or group name is not in the environment, the name is not trusted.

Trusted users and groups (UNIX and Linux)

A trusted user or a trusted group is a user or group that you empower with administering the database server and other important systems.

Trusted users

To run Informix securely, you must trust the following users on your host computer:

root
The host environment is not secure unless you can trust anyone who has been legitimately designated a superuser.

bin and sys
Some environments have these user accounts set up to own programs in system directories such as `/bin` and `/usr/lib` when the owner is not **root**.

informix
The database server is not secure unless you can trust anyone who has been legitimately given the most authoritative privileges over an Informix instance.

Trusted groups

You must also trust the following groups:

- Group **informix**

  Because group **informix** must have read and write permissions on the chunk files that hold data, any user in this group can read or modify any unencrypted data in a database. The only user that belongs to group **informix** is user **informix**.
- Group ID 0 (zero)

  This group typically has authority over many key directories. The name of the account with group ID 0 varies across operating systems: group **root**, group **wheel** or group **system**. On Mac OS X, group **admin** (group ID 80) has authority over the root directory.
- Groups **bin** and **sys** (when present)

  These groups typically administer system files and directories that do not belong to group **root**.

Secure directory permissions (UNIX and Linux)

The installation directory and its subdirectories require specific permissions, depending on each directory's function.

Table 1-1. Secure permissions for the installation directory and subdirectories

Subdirectory	Owner	Group	Permissions (octal)
`. ($INFORMIXDIR)`	informix	informix	755

Table 1-1. Secure permissions for the installation directory and subdirectories (continued)

Subdirectory	Owner	Group	Permissions (octal)
bin	informix	informix	755
lib	informix	informix	755
gls	informix	informix	755
msg	informix	informix	755
etc	informix	DBSA	775
aaodir	informix	AAO	775
dbssodir	informix	DBSSO	775
tmp	informix	informix	770

See "Administrative roles and role separation" on page 8-1 for more information about Database Server Administrator (DBSA), Audit Analysis Officer (AAO), and Database System Security Officer (DBSSO) groups.

The onsecurity utility (UNIX and Linux)

The **onsecurity** utility checks the security of a file, directory, or path. It also troubleshoots the security problems if any are detected.

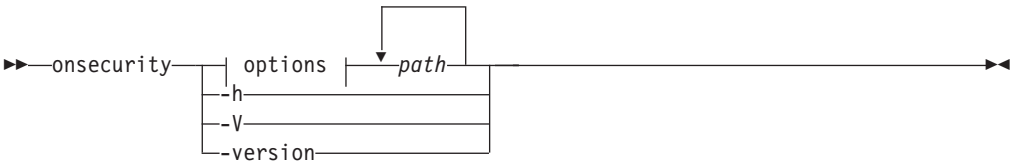
Purpose

Use the **onsecurity** command for one or more of the following purposes:

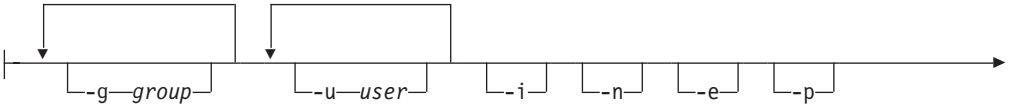
- Check whether a path leading to a directory or a file is secure.
- Generate diagnostic output that explains the nature of the security problem.
- Generate a script that can be run user **root** to remedy the security problems. You can use the script as generated or modify it to meet your environment's security requirements.
- For special circumstances only, specify that particular users, groups, or directories that are normally not trusted can be trusted by the Informix utilities. Add the information to files in the `/etc/informix` directory.

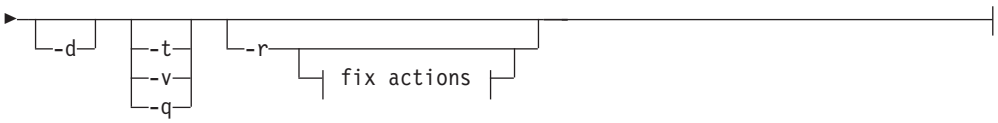
Most frequently, when you run the command on an Informix installation path, you receive a message that the path is secure. If the path is secure, you are not required to do any further work with the utility for the path.

Syntax

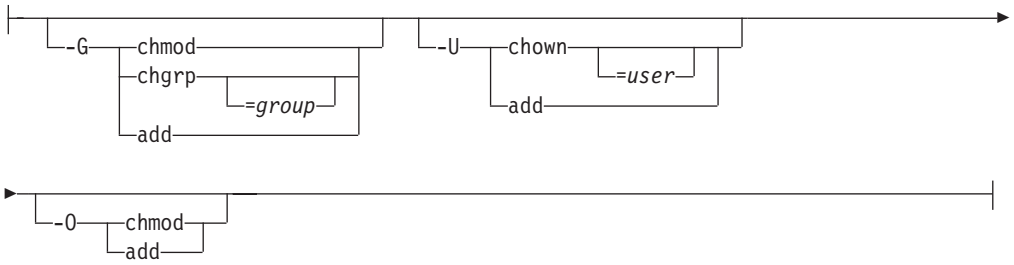


Options:





Fix actions:



Parameters

The following table identifies the syntax terms of the **onsecurity** syntax diagram.

Element	Purpose	Key Considerations
path	Specifies the directory or file path that the utility analyzes.	
group	Specifies a group name or a group number.	
user	Specifies a user name or user number.	

The following table describes valid options for the **onsecurity** command.

Element	Purpose	Key Considerations
`-d`	Prints debugging information.	Implies the `-v` option.
`-h`	Prints a help message listing the supported options and their functions.	
`-V`	Prints short version information and exits the command-line utility.	
`-version`	Prints extended version information and exits the command-line utility.	
`-t`	Prints a terse analysis of the path only if a security problem is detected.	
`-v`	Prints a verbose analysis of the path, regardless of whether a security problem is detected .	

Element	Purpose	Key Considerations
-q	Runs the command in quiet mode. The command prints no information but just exits with a status of either 0 (all paths are secure) or 1 (at least one part of a path is not secure).	No analysis of the security condition is displayed when you use this option, even if the path is not secure (status of 1).
-r	Generates recommendation about how to fix security problems on the path, if there are any.	If the utility detects a security problem in the path, it prints a diagnosis of the problem in a shell script that user **root** can run to fix the security problem. Review the suggested remedy before running the script.
-g *group*	Designates the specified group as trusted for this run of the **onsecurity** command. Other utilities do not trust this group. A group specified by this option is not added to the list of trusted groups in the /etc/informix subdirectory.	If the specified group is already a trusted group, this option has no effect on the diagnostic output or the generated script.
-u *user*	Designates the specified user as trusted for this run of the **onsecurity** command. Other utilities do not trust this user. A user specified by this option is not added to the list of trusted users in the /etc/informix subdirectory.	If the specified user is already a trusted user, this option has no effect on the diagnostic output or the generated script.
-i	Directs the **onsecurity** command to process directories belonging to user and group **informix** as not trusted.	This option is generally more useful in checking the path security of non-Informix software.
-n	Directs the **onsecurity** command to process directories belonging to a system user or system group, such as sys or bin, as not trusted.	
-e	Directs the **onsecurity** command to not check files in /etc/informix.	
-p	Runs the **onsecurity** command in a mode that is appropriate for non-root installations.	When you run the command with the -p option on a path to a non-root installation, you are adding your user login name to the list of trusted users. Also, when you run the command, this option: • Processes directories belonging to user and group **informix** as not trusted. • Excludes files in /etc/informix from the security check.
-G *fix action*	Configure the security script that **onsecurity** generates so that directories with nonsecure group permissions are set as indicated by the specified action.	If you do not specify the -G option, the command assumes that you intended to specify -G chmod.

Element	Purpose	Key Considerations
-U *fix action*	Configure the security script that **onsecurity** generates so that directories with nonsecure user permissions are set as indicated by the specified action.	If you do not specify the -U option, the command assumes that you intended to specify -U chown.
-O *fix action*	Configure the security script that **onsecurity** generates so that directories with nonsecure write access settings are set as indicated by the specified action.	If you do not specify the -O option, the command assumes that you intended to specify -O chmod.
chgrp [=*group*]	Changes the current group to the group that you specify.	If you do not specify a group, changes the group to group 0 (which is called **root**, **wheel**, or **system**, depending on your operating system).
chown [=*user*]	Changes the current owner to the user that you specify as a fix action.	If you do not specify a user, changes the owner to user **root**.
chmod	Removes write access of the group or user on directories, depending on whether the -G or -O option is invoked prior.	
add	• With -G option: Adds current nonsecure group assigned to directory to the /etc/informix/trusted.gids file • With -U option: Adds current nonsecure owner of directory to the /etc/informix/trusted.uids file • With -O option: Adds nonsecure directories to the etc/informix/trusted.insecure.directories file	**Important:** Use the add option in the **onsecurity** command only if there is no acceptable alternative. **onsecurity -O add** is particularly hazardous if you are not vigilant about the security of your system after running the command. You must not use the -O add option.

Usage

When the **onsecurity** utility detects a problem, it is crucial that you fix the problem before running any of the other Informix utilities because they will exit reporting the same problem. Use the -r option to view the recommended actions to correct detected security flaws. If after reading the diagnostic output you realize that you want to configure the script to override the database server's security mechanisms to allow certain nonsecure users, groups, or directory permissions in the installation path, you can use the -r option with -G, -U, or -O.

When you use the -r option, a script is written to standard output that would fix security problems. The script is not run by the **onsecurity** utility. A user who has root privileges must review the proposed fix before running the script. The script cannot be run by a user who does not have root privileges.

To run the **onsecurity** utility so that it does not flag a specific group or specific user as a security problem, you can use the -g and -u options. For example, if you added -g 8714 or -g ccusers to the command line, the **onsecurity** utility would not report that the group is untrusted.

The -g and -u options do not change any directory settings and do not change what constitutes secure settings for the database server. These options affect only the diagnostic output of **onsecurity**; not the trusted entities in the /etc/informix/ subdirectories and not the script generated with the -r option.

The -p option is only useful for checking the security of a non-root installation path. This option implicitly has the properties of the -i, -e, and -u options.

Examples

The following example shows the output from running the **onsecurity** utility on a path that is secure:

```
$ onsecurity /usr/informix/11.50.FC4
# /usr/informix/11.50.FC4 resolves to /work4/informix/Operational/11.50.FC4
(path is trusted)
```

In the preceding example, the specified path /usr/informix/11.50.FC4 traverses at least one symbolic link to end up at the actual directory /work4/informix/Operational/11.50.FC4, but the whole path is secure.

The following example shows the output from running **onsecurity** on a path that is not secure:

```
$ onsecurity /work/informix/ids-11

# !!! SECURITY PROBLEM !!!
# /work/informix/ids-11 (path is not trusted)
# Analysis:
# User           Group          Mode Type Secure Name
# 0     root     0     root     0755 DIR    YES  /
# 0     root     0     root     0755 DIR    YES  /work
# 203   unknown  8714  ccusers  0777 DIR    NO   /work/informix
# 200   informix 102   informix 0755 DIR    NO   /work/informix/ids-11
# Name: /work/informix
# Problem: owner <unknown> (uid 203) is not trusted
# Problem: group ccusers (gid 8714) is not trusted but can modify the directory
# Problem: the permissions 0777 include public write access
```

In the preceding example, the informix directory of the path /work/informix has the following security flaws:

- the owner of this directory is not a trusted user
- the group that controls the directory is not trusted
- the directory has public write access

Securing a nonsecure $INFORMIXDIR and its subdirectories (UNIX and Linux)

Run the **make-informixdir-secure** script and follow messages that it displays when a running database server is no longer secure.

To secure $INFORMIXDIR and subdirectories when the running database server detects a problem:

As user **root**, run the **$INFORMIXDIR/etc/make-informixdir-secure** script.

Although user **informix** has permission to run the script, the script cannot fix the problems unless the directory is owned by user **informix**. The database server message indicates what still must be fixed. The script also shows files and directories under $INFORMIXDIR that belong to an unexpected owner or group or have public write permission.

Disabling the security check of INFORMIXDIR and subdirectories

You must never disable security checking on INFORMIXDIR, but you can partially disable the automatic security check of a specific installation directory.

This task is intended only if you have no other recourse in order to do essential work on the database server and can accept the consequences of disabling security on INFORMIXDIR. If you disable the security checking, you must use the **ibmifmx_security.sh** script to limit the number of SUID and SGID programs on your system.

Important: The following script causes Informix to run with an INFORMIXDIR that has public write access, which can open up your system to security breaches.

To disable security checking:

As the user **root**, run the **INFORMIXDIR/etc/informixdir-is-insecure** script. After this script runs successfully, the warning messages still open when the utilities are run, but the programs continue. You can specify the value of INFORMIXDIR on the command line as an argument to the script. Thus, you are not required to set INFORMIXDIR in the **root** user environment.

The **informixdir-is-insecure** script creates a /etc/informix directory (if necessary) that is owned by **root** and has 555 permissions. In this directory, the script creates a file named server-*xx.xx.yyy* that has 444 permissions. The *xx.xx* portion of the file name is the major version number and *yyy* portion is the fix pack number: for example, server-11.70.UC1. This file lists the $INFORMIXDIR values for which security checking is disabled.

If you later upgrade Informix, you will be prompted to verify that you want to continue using an INFORMIXDIR that is not secure in the newer version.

Security warnings and error messages at server startup (UNIX and Linux)

If a security check that a server utility performs at startup detects a problem, the security check returns an error message or warning.

These messages are returned when the message file and internationalization support are unavailable. Therefore, the error messages do not have error numbers and are not translated.

The following list shows security-related messages that can open when startup of the database server is attempted. In most environments, the server utility automatically exits when it detects one of these problems.

- INFORMIXDIR or ONCONFIG is too long. Maximum length for $INFORMIXDIR/etc/$ONCONFIG is 255 characters.
- INFORMIXSQLHOSTS is too long. Maximum length is 255 characters.
- TBCONFIG is not supported and will not be used.
- User informix not found.

- `Group informix not found.`
- `Could not access logical-file file name.`
- `Logical-file file name is not owned by user with id UID.`
- `Logical-file file name not owned by group with id GID.`
- `Logical-file file name has insecure mode mode.`

The following table defines the variables used in the preceding messages.

Variable	Explanation
file name	A name of the file or directory
logical-file	onconfig, INFORMIXSQLHOSTS, `INFORMIXDIR`, or `INFORMIXDIR/xxx` (where *xxx* is one of a number of subdirectories under `$INFORMIXDIR`). For example, if the INFORMIXDIR environment variable is set to `/usr/informix`, the message might read: `INFORMIXSQLHOSTS /usr/informix/etc/sqlhosts is not owned by the user with id 1234.`
mode	An octal permissions value
UID	A user ID
GID	A group ID

Users and group membership for running utilities

Most IBM Informix utilities run as secure users and belong to a secure group.

The following database server utilities are SUID **root** and SGID **informix**:

- **onaudit**
- **onbar_d**
- **ondblog**
- **onedcu**
- **oninit**
- **onmode**
- **ON-Monitor**
- **onshowaudit**
- **onsmsync**
- **onsnmp**
- **onsrvapd**
- **ontape**
- **snmpdm**

The following database server utilities are SGID **informix**:

- **oncheck**
- **onedpdu**
- **onload**
- **onlog**
- **onparams**
- **onpload**
- **onspaces**
- **onstat**

- **onunload**
- **xtree**

UNIX and Linux only: The previous utilities do not run if the installation path is not secure. This is a security precaution to help prevent tampering with your Informix installation.

Restriction: You cannot use the following utilities on HDR secondary servers, remote stand-alone (RS) secondary servers, or shared disk (SD) secondary servers:

- **archecker**
- **HPL**
- **dbimport**
- **dbexport**
- **dbload**
- **onaudit**
- **ondblog**
- **onload**
- **onmonitor**
- **onparams**
- **onperf**
- **onshowaudit**
- **onsnmp**
- **onspaces**
- **onunload**

Security of the chunk files

For Informix security, store data in chunk files that are owned by user **informix**, belong to group **informix**, and have 660 permissions.

The directory holding the chunk files must be secure, following the same rules as those that ensure the installation directory is secure. Similarly, all other files and directories configured for use by Informix must be secure.

You can use the **onsecurity** utility to check if there are security problems with the directory holding the chunk files. The utility prints a diagnosis of any such problems, and can suggest a way to fix them.

Do not use `/tmp` as the directory for any log files or dump files. However, it is generally safe to create and use a subdirectory such as `/tmp/informix` if the subdirectory has appropriately restricted permissions. Typically, a subdirectory like `/tmp/informix` is owned by user and group **informix** and does not have any public access permissions.

Security for loading external modules

Use the **DB_LIBRARY_PATH** configuration parameter to control the location from which shared objects, such as external modules, can be loaded.

Use the **DB_LIBRARY_PATH** configuration parameter to specify a comma-separated list of valid directory prefix locations from which the database server can load

external modules, such as DataBlade modules. **DB_LIBRARY_PATH** takes effect when the database server is restarted after the parameter has been set.

Use the **DB_LIBRARY_PATH** configuration parameter to control the location from which shared objects can be loaded, and enforce policies and standards on the formats of the EXTERNAL NAME clause of the CREATE FUNCTION, CREATE PROCEDURE, and CREATE ROUTINE statements.

If the **DB_LIBRARY_PATH** configuration parameter is not set or is not present in the `onconfig` file, security checks for loading external modules are not performed.

You must include in the **DB_LIBRARY_PATH** settings every file system in which your security policy authorizes DataBlade modules and UDRs to be located. DataBlade modules provided with IBM Informix are stored under the `$INFORMIXDIR/extend` directory. For extensibility to work properly when security is turned on, the string "`$INFORMIXDIR/extend`" must be part of **DB_LIBRARY_PATH**.

For more information about the **DB_LIBRARY_PATH** configuration parameter, see the *IBM Informix Administrator's Reference*.

Chapter 2. Network data encryption

Use network encryption to encrypt data transmitted between server and client, and between server and other server.

Encryption is the process of transforming data into an unintelligible form to prevent the unauthorized use of the data. To read an encrypted file, you must have access to a secret decryption key or password. Unencrypted data is called *plain text*; encrypted data is called *cipher text*. A *cipher* is an encryption-decryption algorithm.

Communication support modules for data transmission encryption

You can use the communication support modules (CSMs) to encrypt data transmissions, including distributed queries, over the network.

The encryption CSM (ENCCSM) provides network transmission encryption.

This option provides complete data encryption with a standard cryptography library, with many configurable options. A message authentication code (MAC) is transmitted as part of the encrypted data transmission to ensure data integrity. A MAC is an encrypted message digest.

CSMs have the following restrictions:

- You cannot use an encryption CSM and a simple password CSM simultaneously. For example, if you are using the simple password CSM, SPWDCSM, and decide to encrypt your network data, you must remove the entries for the SPWDCSM in your `concsm.cfg` and `sqlhosts` files.
- You cannot use either simple password CSM or encryption CSM over a multiplexed connection.
- Enterprise Replication and high-availability clusters (High-Availability Data Replication, remote stand-alone secondary servers, and shared disk secondary servers) support encryption, but cannot use a connection configured with a CSM. See "Enterprise replication and high availability network data encryption" on page 2-10 for more information about this topic.
- Encrypted connections and unencrypted connections cannot be combined on the same port.

Secure Sockets Layer (SSL) communications, which encrypt data in end-to-end, secure TCP/IP and Distributed Relational Database Architecture™ (DRDA®) connections between two points over a network, are an alternative to the IBM Informix-specific encryption CSMs. For more information, see "Secure sockets layer protocol" on page 2-11.

Enabling encryption with communication support modules

You must modify the `concsm.cfg` file to use encryption with communication support modules.

Verify that the module can use a port that is not shared with an unencrypted connection before you enable network encryption.

To enable network encryption:

1. Add a line to the concsm.cfg file. The concsm.cfg file must contain an entry for each communications support module (of the same kind) that you are using.
2. Add an entry to the options column of the sqlhosts file or registry. For information about specifying the CSM in the sqlhosts file or registry, see the *IBM Informix Administrator's Guide*.

CSM configuration file

To use a communication support module (CSM), you must have a concsm.cfg file.

An entry in the concsm.cfg file is a single line and is limited to 1024 bytes. After you describe the CSM in the concsm.cfg file, you can enable it in the **options** parameter of the sqlhosts file, as described in *IBM Informix Administrator's Guide*.

The concsm.cfg file is located in the etc directory of INFORMIXDIR by default. If you want to store the file somewhere else, you can override the default location by setting the INFORMIXCONCSMCFG environment variable to the full path name of the new location. For information about setting the environment variable INFORMIXCONCSMCFG, see the *IBM Informix Guide to SQL: Reference*.

Entries in the concsm.cfg file must conform to the following restrictions:

- The following characters are not allowed to be part of library path names:
  - = (equal sign)
  - " (double quotation mark)
  - , (comma)
- White spaces cannot be used unless the white spaces are part of a path name.

Encryption ciphers and modes

You must specify which ciphers and mode to use during encryption.

The cipher and mode that is used is randomly selected among the ciphers that are common between the two servers. Make sure that all servers and client computers that participate in encrypted communication have ciphers and modes in common. Encryption is more secure if you include more ciphers and modes that the database server can switch between. For information about how to switch between ciphers, see "Switch frequency" on page 2-5.

The Data Encryption Standard (DES) is a cryptographic algorithm designed to encrypt and decrypt data by using 8-byte blocks and a 64-bit key.

The Triple DES (DES3) is a variation of DES in which three 64-bit keys are used for a 192-bit key. DES3 works by first encrypting the plain text by using the first 64-bits of the key. Then the cipher text is decrypted by using the next part of the key. In the final step, the resulting cipher text is re-encrypted by using the last part of the key.

The Advanced Encryption Standard (AES) is a replacement algorithm that is used by the United States government.

Two encryption modes are:

- *Block Mode*, a method of encryption in which the message is broken into blocks and the encryption occurs on each block as a unit. Since each block is at least 8 bytes large, block mode provides the ability for 64-bit arithmetic in the encryption algorithm.

- *Stream Mode*, a method of encryption in which each individual byte is encrypted. It is generally considered to be a weak form of encryption.

A *Blowfish* is a block cipher that operates on 64-bit (8-byte) blocks of data. It uses a variable size key, but typically, 128-bit (16-byte) keys are considered to be good for strong encryption. Blowfish can be used in the same modes as DES.

Important: You must not specify individual ciphers. For security reasons, all ciphers must be allowed. If a cipher is discovered to have a weakness, you can exclude it.

Use the `allbut` option to list ciphers and modes to exclude. Enclose the `allbut` list in angled brackets (<>). The list can include unique, abbreviated entries. For example, **bf** can represent **bf1**, **bf2**, and **bf3**. However, if the abbreviation is the name of an actual cipher, then only that cipher is eliminated. Therefore, **des** eliminates only the DES cipher, but **de** eliminates **des**, **ede**, and **desx**.

The following **des**, **ede**, and **desx** ciphers are supported.

Cipher	Explanation	Blowfish Cipher	Explanation
des	DES (64-bit key)	bf1	Blowfish (64-bit key)
ede	Triple DES	bf2	Blowfish (128-bit key)
desx	Extended DES (128-bit key)	bf3	Blowfish (192-bit key)

Important: The cipher **desx** can only be used in **cbc** mode.

The following AES-encryption ciphers are supported.

Cipher	Explanation
aes	AES (128-bit key)
aes128	AES (128-bit key)
aes192	AES (192-bit key)
aes256	AES (256-bit key)

The following modes are supported.

Mode	Explanation
ecb	Electronic Code Book
cbc	Cipher Block Chaining
cfb	Cipher Feedback
ofb	Output Feedback

Because **ecb** mode is considered weak; it is only included if specifically requested. It is not included in the `all` or the `allbut` list.

Related reference:

"Specifying network encryption options in concsm.cfg" on page 2-5

MAC key files

The MAC key files contain encryption keys that are used to encrypt messages.

The database servers and client computers that participate in encryption normally require the same MAC key file. For information about how to switch between MAC keys, see "Switch frequency" on page 2-5.

The default MAC key file is the built-in file provided by IBM Informix. This file provides limited message verification (some validation of the received message and determination that it has come from an IBM Informix client or server). A site-generated MAC key file performs the strongest verification. You can generate key files with the **GenMacKey** utility.

Each of the MAC key files is prioritized and negotiated at connect time. The prioritization for the MAC key files is based on their creation time by the **GenMacKey** utility. The built-in key file has the lowest priority.

Tip: If there are no MAC key files present, the built-in MAC key is used by default. However, by using a MAC key file, the default built-in MAC key is disabled.

Related reference:

"Specifying network encryption options in concsm.cfg" on page 2-5

Generating a new MAC key file

You can generate a new MAC key file to improve the reliability of message verification using encryption.

To generate a new MAC key file:

1. Run the following command from the command line:

   `GenMacKey –o` *`filename`*

   The *`filename`* is the path and file name of the new MAC key file.
2. Update the central server's configuration to include the location of the new MAC key file in one of the following ways:
   - **Using encryption tags:** Edit the relevant line in the `concsm.cfg` file to add a path and file name to the mac tag.
   - **Using encryption parameters:** Edit the encryption parameters file to alter the value of the ENCCSM_MACFILES parameter.
3. If necessary, remove old MAC key file entries from the configuration.
4. Distribute the new MAC key file among all appropriate computers.

Related reference:

"Specifying network encryption options in concsm.cfg" on page 2-5
"ENCCSM_MACFILES parameter" on page 2-10

MAC levels

MAC levels determine the type of MAC key generation.

The supported generation levels are:

- `high`. Uses SHA1 MAC generation on all messages.

- `medium`. Uses SHA1 MAC generation for all messages greater than 20 bytes long and XOR folding on smaller messages.
- `low`. Uses XOR folding on all messages.
- `off`. Does not use MAC generation.

The level is prioritized to the highest value. The `off` entry must only be used between servers when it is guaranteed that there is a secure network connection.

All servers and client computers that transmit encrypted communication must have at least one MAC level setting in common. For example, if one database server has a level of `high` and `medium` enabled and the other database server has only `low` enabled, then the connection attempt fails. But if a database server has `high` and `medium` settings and the other database server has only the `medium` setting, the MAC generation levels support a connection.

Related reference:

"Specifying network encryption options in concsm.cfg"

Switch frequency

The switch frequency defines when ciphers and or secret keys are renegotiated.

The default time that this renegotiation occurs is once an hour. By using switch options, you can set the time in minutes when the renegotiation occurs.

The longer that the secret key and encryption cipher remain in use, the more likely that the encryption rules might be broken by an attacker. To avoid this, cryptologists recommend periodically changing the secret key and cipher on long-term connections.

Network data encryption syntax

You must specify network encryption libraries and options in the `concsm.cfg` file.

You can specify the following types of encryption options:

- DES and AES ciphers to use during encryption
- Modes to use during encryption
- Message authentication code (MAC) key files
- MAC levels
- Switch frequency for ciphers and keys

You can use the following methods to specify encryption options.

- "Invoking an encryption parameters file in concsm.cfg" on page 2-8
- "Specifying network encryption options in concsm.cfg"

Note: In `concsm.cfg`, invoking an encryption parameters file is simpler than using encryption tags.

Specifying network encryption options in concsm.cfg

You can modify encryption communication support module (CSM) options by specifying libraries and encryption tags.

IBM Informix provides the following shared libraries for use as CSMs. The paths and fixed file names are:

- `$INFORMIXDIR/lib/client/csm/iencs11a.so` (UNIX and Linux)

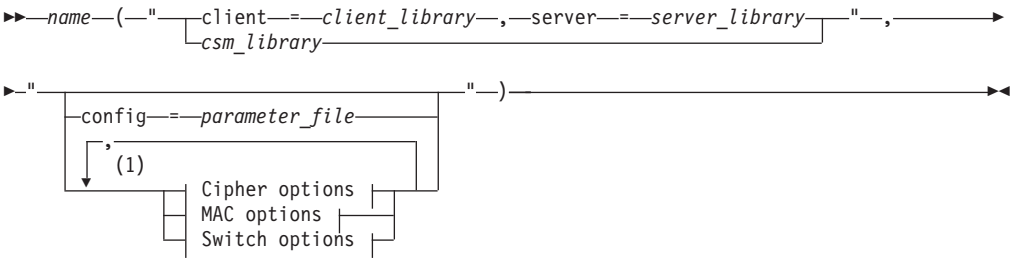
- %INFORMIXDIR%\bin\client\iencs11a.dll (Windows)

The shared libraries also have version-specific names that can be used in place of the fixed names. If you use the version-specific name, and the server is updated, you must update the concsm.cfg file.

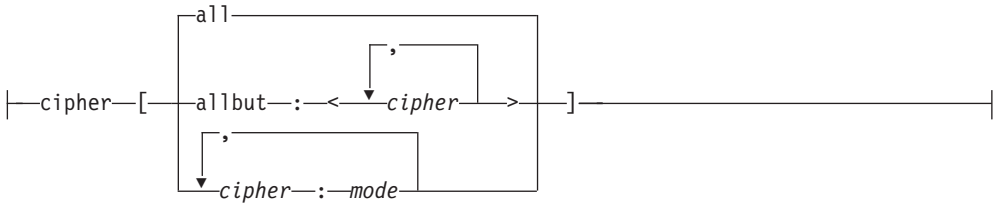
Note: Specifying encryption options directly in the concsm.cfg file is usually more difficult than specifying libraries and tags in an encryption parameters file because of syntax specifications. A sample file concsm.example is available in $INFORMIXDIR/etc (UNIX and Linux).

To configure the CSM for network encryption, use the following syntax to add a line to $INFORMIXDIR/etc/concsm.cfg (UNIX and Linux) or %INFORMIXDIR%\etc\concsm.cfg (Windows).

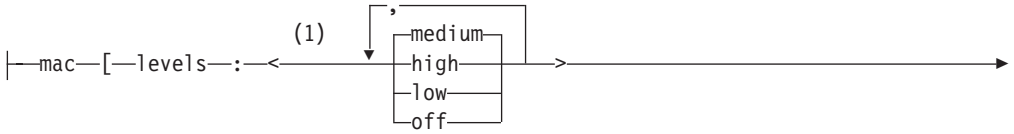
concsm.cfg entry Syntax



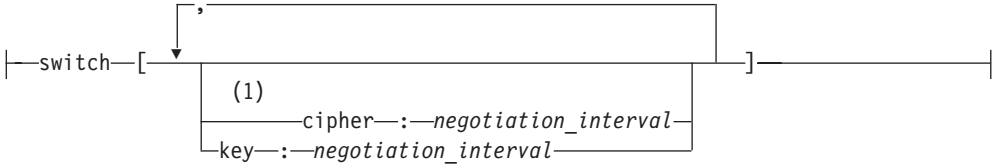
Cipher options:



MAC options:



Switch options:



Notes:

1 Use each path no more than once.

Option	Description
`all`	Include all available ciphers and all available modes, except ECB mode.
`allbut`	Include all ciphers except the ones listed.
`builtin`	The default MAC key file provided by IBM Informix. The builtin file provides limited message verification that received messages have come from an IBM Informix client or server).
cipher	Include the specified cipher.
client_library	The path and name of the shared library that is the CSM on the client computer.
csm_library	The path and name of the shared library that is the CSM if the CSM is shared by both the database server and the client computers.
`files`	The comma-separated list of the full path names of MAC key files.
`key`	Message authentication code (MAC) keys used for message encryption.
key_file	The path and file name of the MAC key files.
`levels`	Specifies a comma-separated list of MAC generation levels that the connection supports. **high** Use SHA1 MAC generation on all messages. **medium** Use SHA1 MAC generation for all messages greater than 20 bytes long and XOR folding on smaller messages. **low** Use XOR folding on all messages. **off** Do not use MAC generation.
mode	Use the specified cipher mode. **ecb** Electronic Code Book **cbc** Cipher Block Chaining **cfb** Cipher Feedback **ofb** Output Feedback
name	The name that you assign to the CSM.
negotiation_interval	The minutes between renegotiations.

Option	Description
parameter_file	The path and file name of the file in which the encryption parameters are defined. **Important:** If the file does not exist at the specified path, then default parameter values are used. No error is returned.
server_library	The full path and name of the shared library that is the CSM on the database server.

Examples of using encryption tags

The following configuration string states to use all available ciphers except for any of the Blowfish ciphers, and to not use any cipher in ECB mode:

```
ENCCSM("$INFORMIXDIR/lib/csm/iencs11a.so",
 "cipher[allbut:<ecb,bf>]")
```

The following configuration string states:

- Use the DES/CBC-mode, EDE/OFB-mode, and DESX/CBC-mode ciphers for this connection.
- Use either SHA1 MAC generation or XOR folding on all messages.
- Use `mac1.dat`, mac2.dat, or the builtin MAC key file for encrypting messages.
- Switch the cipher being used every 120 minutes and renegotiate the secret key every 15 minutes.

```
ENCCSM("/$INFORMIXDIR/lib/csm/iencs11a.so",
   "cipher[des:cbc,ede:ofb,desx:cbc],
   mac[levels:<high,low>,files:</usr/local/bin/mac1.dat,
   /usr/local/bin/mac2.dat,builtin>],
   switch[cipher:120,key:15]")
```

Related concepts:

"MAC key files" on page 2-4

"Encryption ciphers and modes" on page 2-2

"MAC levels" on page 2-4

Related tasks:

"Generating a new MAC key file" on page 2-4

Related reference:

"Invoking an encryption parameters file in concsm.cfg"

Invoking an encryption parameters file in concsm.cfg

You can configure encryption options by setting encryption parameters in a file and then invoking it in the `concsm.cfg` file.

In the encryption parameters file that you specify in the `concsm.cfg` file, each option has the following form:

```
PARAMETER_NAME value
```

Use the following parameters to set encryption options:

- **ENCCSM_CIPHERS**: *Ciphers to be used*
- **ENCCSM_MAC**: *MAC levels*
- **ENCCSM_MACFILES**: *MAC file locations*
- **ENCCSM_SWITCH**: *Cipher and key change frequency*

The following rules apply to the parameter values:

- Each entry must be of the form *PARAMATER_NAME value* separated by white spaces (for example, `ENCCSM_MAC medium,high` and `ENCCSM_MACFILES /usr/local/bin/mac1.dat,/usr/local/bin/mac2.dat,builtin`).

  Note: White spaces are not allowed within a value.
- Each parameter must have one entry in the configuration file. If multiple entries exist, only the first entry is used.
- Default values are used if a parameter does not exist in the configuration file.
- Characters after a comment character (#) are ignored; however, the path name value is not ignored.

Related reference:

"Specifying network encryption options in concsm.cfg" on page 2-5

ENCCSM_CIPHERS parameter:

The **ENCCSM_CIPHERS** parameter specifies the ciphers and modes to use during encryption.

```
syntax ENCCSM_CIPHERS

all|allbut:<list of ciphers and
modes>|cipher:mode{,cipher:mode ...}
```

- `all`: Specifies to include all available ciphers and modes, except ECB mode. For example:

  ```
  ENCCSM_CIPHERS all
  ```
- `allbut:<`*`list of ciphers and modes`*`>`: Specifies to include all ciphers and modes except the ones in the list. Separate ciphers or modes with a comma. For example:

  ```
  ENCCSM_CIPHERS allbut:<cbc,bf>
  ```
- `cipher:`*`mode`*: Specifies the ciphers and modes. Separate cipher-mode pairs with a comma. For example:

  ```
  ENCCSM_CIPHERS des3:cbc,des3:ofb
  ```
- `default value: allbut:<ecb>`

For more information about ciphers and modes, see "Encryption ciphers and modes" on page 2-2.

ENCCSM_MAC parameter:

The **ENCCSM_MAC** parameter specifies the MAC level to use.

default value

`medium`

range of values

One or more of the following options, separated by commas:

- `off` does not use MAC generation.
- `low` uses XOR folding on all messages.
- `medium` uses SHA1 MAC generation for all messages greater than 20 bytes long and XOR folding on smaller messages.
- `high` uses SHA1 MAC generation on all messages.

For example: `ENCCSM_MAC medium,high`

For more information about MAC levels, see “MAC levels” on page 2-4.

ENCCSM_MACFILES parameter:

The **ENCCSM_MACFILES** parameter specifies the MAC key files to use.

default value
`builtin`

units Path names, up to 1536 bytes in length

range of values
One or more full path and file names separated by commas, and the optional `builtin` keyword. For example: `ENCCSM_MACFILES /usr/local/bin/mac1.dat,/usr/local/bin/mac2.dat,builtin`

For more information, see “MAC key files” on page 2-4.

Related tasks:

“Generating a new MAC key file” on page 2-4

ENCCSM_SWITCH parameter:

The **ENCCSM_SWITCH** parameter defines the number of minutes between cipher and key renegotiation.

syntax ENCCSM_SWITCH *cipher_switch_time,key_switch_time*

- *cipher_switch_time* specifies the minutes between cipher renegotiation
- *key_switch_time* specifies the minutes between secret key renegotiation

default value
`60,60`

units minutes

range of values
positive integers

For more information, see “Switch frequency” on page 2-5.

Example of encryption parameter file:

The encryption parameter file specifies values for encryption parameters.

The following example shows an encryption parameter file:

```
ENCCSM_CIPHERS         all
ENCCSM_SWITCH          120,60
ENCCSM_MAC             medium
ENCCSM_MACFILES     /usr/informix/etc/MacKey.dat
```

The following example illustrates a line in the `concsm.cfg` file to specify encryption with a parameter file named `encrypt.txt`:

```
ENCCSM("usr/informix/lib/cms/iencs11a.so",
"config=/usr/lib/encrypt.txt")
```

Enterprise replication and high availability network data encryption

You can configure network data encryption for Enterprise Replication and high availability clusters by using configuration parameters.

Important: You cannot start Enterprise Replication or high availability options on a network connection that is configured to use communication support module (CSM) encryption for client/server connections. CSM encryption must be configured to use a separate network port.

You can use Enterprise Replication and high availability encryption parameters to encrypt the data traffic between the servers participating in Enterprise Replication and high availability clusters (High-Availability Data Replication, remote stand-alone secondary servers, and shared disk secondary servers). High availability encryption works with Enterprise Replication encryption and each operates whether the other is enabled or not.

The following configuration parameters configure encryption for Enterprise Replication and high availability clusters:

- **ENCRYPT_CIPHERS**: defines all ciphers and modes that can be used by the current database session
- **ENCRYPT_MAC**: controls the level of message authentication code (MAC) generation
- **ENCRYPT_MACFILE**: specifies a list of the full path names of MAC key files
- **ENCRYPT_SWITCH**: defines the frequency at which ciphers or secret keys are renegotiated
- **ENCRYPT_CDR**: sets the level of encryption for Enterprise Replication
- **ENCRYPT_HDR**: enables or disables HDR encryption
- **ENCRYPT_SMX**: sets the level of encryption for remote stand-alone and shared disk secondary servers

When working with each other, high availability and Enterprise Replication share the same **ENCRYPT_CIPHERS**, **ENCRYPT_MAC**, **ENCRYPT_MACFILE** and **ENCRYPT_SWITCH** configuration parameters.

While an encrypted high availability or Enterprise Replication connection operates from server to server, CSM network encryption operates between client and server. Both types of encryption can run on the same network if configured as follows:

- One network port must be configured for high availability.
- The other network port must be configured for CSM connections.

For information about these configuration parameters, see *IBM Informix Administrator's Reference*.

Secure sockets layer protocol

The Secure Sockets Layer (SSL) protocol is a communication protocol that uses encryption to provide privacy and integrity for data communication through a reliable end-to-end secure connection between two points over a network.

You can use SSL for the following connections:

- IBM Data Server Driver for JDBC and SQLJ connections with Informix
- IBM Informix ESQL/C connections with Informix
- IBM Informix ODBC Driver connections with Informix
- DB-Access connections
- Enterprise Replication connections

- High-availability data replication (HDR) connections between an HDR primary server and one or more secondary servers of any type (HDR secondary, SD secondary, or RS secondary)
- Distributed transaction connections, which span multiple database servers
- The **dbexport**, **dbimport**, **dbschema**, and **dbload** utility connections
- Connection Manager connections between servers in a cluster

The SSL protocol provides these advantages over the Informix communication support modules (CSMs):

- SSL is a more widely used alternative to the IBM Informix CSMs.
- You can use SSL for encrypted communication with both DRDA and SQLI clients. You can use the CSMs only for connections with SQLI clients; you cannot use them for connections with DRDA clients.

You can also configure the Encrypt and Simple Password Communications Support Modules (ENCCSM and SPWDCSM) with SSL connections. However, because these CSMs provide encryption functionality, configuring the ENCCSM or SPWDCSM with SSL involves additional effort with no extra benefit.

You can configure Pluggable Authentication Module (PAM) and the Generic Security Services Communications Support Module (GSSCSM), which uses the Kerberos 5 security protocol for single sign-on (SSO) with SSL connections.

For information about tools for setting up SSL on the database server and clients, see "IBM Global Security Kit" on page 2-14.

Digital certificates that exchange keys in SSL connections

SSL uses digital certificates, which are electronic ID cards issued by a trusted party, to exchange keys for encryption and server authentication.

The trusted entity that issues a digital certificate is known as a Certificate Authority (CA).

The CA issues a digital certificate for only a limited time. When the expiration date passes, you must acquire another digital certificate.

With SSL, the data that moves between a client and server is encrypted by a symmetric key (secret or private key) algorithm. An asymmetric key (public key) algorithm is used for the exchange of the secret keys in the symmetric algorithm.

When a client attempts to connect to a secure server, an SSL handshake occurs. The handshake involves the following events:

1. The server sends its digital certificate to the client.
2. The client verifies the validity of the server digital certificate. For this to occur, the client must possess the digital certificate of the CA that issued the server digital certificate.

If the handshake succeeds, these events occur:

1. The client generates a random symmetric key and sends it to the server, in an encrypted form, by using the asymmetric key in the server digital certificate.
2. The server retrieves the symmetric key by decrypting it.

Because the server and the client now know and can use the symmetric key, the server and client encrypt data for the duration of the session.

Keystores that store SSL keys and digital certificates

A keystore is a protected database that stores SSL keys and digital certificates. Both the client and server must have the keystore that stores the digital certificates used in SSL communication.

The server keystore and its configuration

The keystore stores its digital certificate and the root CA certificate of all other servers that Informix is connecting to. The server keystore must be located in the INFORMIXDIR/ssl directory. The name of the keystore file must be *server_name*.kdb, where *server_name* is the value specified in the **DBSERVERNAME** configuration parameter.

Each Informix instance must have its own keystore.

Each certificate in the keystore has a unique label. When you set up Informix to use SSL, you must specify the name of the label of the digital certificate in the **SSL_KEYSTORE_LABEL** configuration parameter in the onconfig file. If you do not specify a label name in the **SSL_KEYSTORE_LABEL** configuration parameter, Informix uses the default certificate in the keystore for SSL communication. Only one certificate in a keystore is the default certificate.

The keystore is protected by a password that Informix must know so that it can retrieve the digital certificate for SSL communications. Informix stores its keystore password in an encrypted form in a stash (.sth) file in the INFORMIXDIR/ssl directory. The name of the keystore stash file must be *server_name*.sth.

The password for the keystore is mandatory, because this password protects the private key for the server.

The permissions on the INFORMIXDIR/ssl/*server_name*.kdb and $INFORMIXDIR/ssl/*server_name*.sth files must be 600, with informix set as both the owner and the group, even though Informix does not enforce these permissions.

The client keystore and its configuration

The keystore on an Informix client stores the root CA certificates of all servers to which the client is connecting. A password for the keystore is optional on the client.

For Informix SQLI clients (ESQL/C, ODBC, DB-Access, and the **dbexport**, **dbimport**, **dbschema**, and **dbload** utilities), the location of the keystore and its stash file is not fixed. Instead, the conssl.cfg file in the $INFORMIXDIR/etc directory specifies the keystore and the stash file for Informix clients.

The following table shows the client configuration parameters that are in the conssl.cfg file.

Table 2-1. Client configuration parameters in the conssl.cfg file

IBM Informix Client Configuration Parameter	Description
SSL_KEYSTORE_FILE	This is the fully qualified file name of the keystore that stores the root CA certificates of all of the servers to which the client connects.
SSL_KEYSTORE_STH	This is the fully qualified file name of the stash file containing the encrypted keystore password.

If a conssl.cfg file does not exist or the **SSL_KEYSTORE_FILE** and **SSL_KEYSTORE_STH** configuration parameters are not set, the client uses $INFORMIXDIR/etc/client.kdb and $INFORMIXDIR/etc/client.sth as the default keystore and keystore stash file names for the client.

Related reference:

GSKIT_VERSION configuration parameter (Administrator's Reference)

IBM Global Security Kit

The IBM Global Security Kit (GSKit) provides libraries and utilities for SSL communication.

GSKit version 8 is installed with Informix 11.70 and IBM Client Software Development Kit (Client SDK) version 3.70. GSKit 8 is used by default; however, if you have another supported, major version of GSKit software installed on your computer, you can set the GSKIT_VERSION configuration parameter so that the database server uses the other version.

GSKit 8 includes the GSKCapiCmd command-line interface for managing keys, certificates, and certificate requests. For more information about this tool, see the *GSKCapiCmd User's Guide* at ftp://ftp.software.ibm.com/software/webserver/appserv/library/v80/GSK_CapiCmd_UserGuide.pdf.

The iKeyman utility and GSKCmd command-line interface can also be used to manage keys, certificates, and certificate requests, but are a part of IBM JRE 1.6 or later, instead of GSKit 8. For information about the iKeyman utility and the GSKCmd command-line interface, see the *IBM Developer Kit and Runtime Environment, iKeyman 8.0 User's Guide* at http://download.boulder.ibm.com/ibmdl/pub/software/dw/jdk/security/60/iKeyman.8.User.Guide.pdf.

Configuring a server instance for secure sockets layer connections

Configure an IBM Informix instance for Secure Sockets Layer (SSL) connections by adding connection information to the sqlhosts file, setting SSL configuration parameters, and configuring the keystore and the digital certificates it stores.

To configure an Informix instance for SSL connections:

1. Update connection information in the sqlhosts file (UNIX) or the SQLHOSTS registry (Windows) to include information about SSL connections. Use the:
   - **onsocssl** protocol for ESQL/C, ODBC, DB-Access, **dbexport** utility, **dbimport** utility, **dbschema** utility, or **dbload** utility connections
   - **drsocssl** protocol for DRDA connections

The following table shows an example of an sqlhosts file configured for both SSL and non-SSL connections.

Table 2-2. Example of sqlhosts file configured for SSL connections

Server Name	Protocol	Host Name	Server Name
sf_on	**onsoctcp**	sanfrancisco	sf_serv
oak_on	**onsocssl**	oakland	oak_serv
sac_on	**drsocssl**	sacramento	sac_serv

For more information about the sqlhosts file and the SQLHOSTS registry, see the *IBM Informix Administrator's Guide*.

2. Update configuration parameters in the onconfig file, as follows:
   a. Specify the name of the label of the server digital certificate in the **SSL_KEYSTORE_LABEL** configuration parameter.

      The label can contain up to 512 bytes. If you do not specify a label name, Informix uses the default certificate in the keystore.

      For example, specify:

      ```
      SSL_KEYSTORE_LABEL sf_ssl
      ```

   b. Configure poll threads for SSL connections by using the **NETTYPE** configuration parameter.

      If you do not configure poll threads, Informix starts one poll thread.

      For the protocol, specify socssl. The protocol format is **iiippp**, where **iii**=[ipc|soc|tli] and **ppp**=[shm|str|tcp|imc|ssl].

      For example, specify:

      ```
      NETTYPE socssl,3,50,NET
      ```

   c. Configure Encrypt Virtual Processors (VPs) for SSL encryption and decryption operations, by using the **VPCLASS** parameter.

      If Encrypt VPs are not configured, Informix starts one Encrypt VP the first time an SSL operation occurs.

      You can also use the **onmode -p** command to add or drop Encrypt VPs when the database server is in online mode.

      Tip: For large systems, configure multiple Encrypt VPs.

3. Set up a keystore and its password stash file and digital certificate by using the iKeyman utility, GSKCmd command-line interface, or GSKCapiCmd command-line interface.

   To use the iKeyman utility and GSKCmd tool, JRE 1.6 or later must be installed. The GSKCapiCmd tool is a part of the GSKit and does not require Java.

   When you create the password, be sure to:

   - Select the option to stash the password to a file.
   - Name the keystore as *servername*.kdb, where *servername* is value of the **DBSERVERNAME** configuration parameter.
   - Create the keystore and its stash file in the INFORMIXDIR/ssl directory.
   - Set the permissions on the INFORMIXDIR/ssl/*server_name*.kdb and $INFORMIXDIR/ssl/*server_name*.sth files to 600, with informix set as both the owner and the group, even though Informix does not enforce these permissions.

   For example, specify:

```
gsk8capicmd -keydb -create -db sf_server.kdb -pw sf_password
  -type cms -stash

gsk8capicmd -cert -create -db sf_server.kdb -pw sf_password
  -label my_ssl_label -size 1024 -default_cert yes
```

Important: If the DBSA configures the database server to use a different version of GSKit, the version-specific **gsk8capicmd** command must be replaced with command from the different GSKit version. For example, **gsk7capicmd**.

For information about the keystore, the password stash file, and digital certifications, see "Secure sockets layer protocol" on page 2-11.

For information about the iKeyman utility, GSKCmd command-line interface, and the GSKCapiCmd command-line interface, see "IBM Global Security Kit" on page 2-14.

If any of the Informix utilities (such as DB-Access) must connect to the server by SSL, you must configure a client keystore for the utility on the server, following the steps in "Configuring a client for SSL connections."

Related reference:

GSKIT_VERSION configuration parameter (Administrator's Reference)

Configuring a client for SSL connections

Configure an ESQL/C, ODBC, DB-Access, **dbexport**, **dbimport**, **dbschema**, or **dbload** connection by adding connection information to the `sqlhosts` file, setting SSL configuration parameters, and configuring the keystore and the digital certificates it stores.

Prerequisite: For general information about Secure Sockets Layer (SSL) client connections, see "Secure sockets layer protocol" on page 2-11.

1. Update connection information in the `sqlhosts` file (UNIX) or the SQLHOSTS registry (Windows), by using the **onsocssl** protocol for SSL SQLI client connections.

   The following table shows an example of an `sqlhosts` file configured for these client connections.

Table 2-3. Example of `sqlhosts` file configured for SSL SQLI client connections

Server Name	Protocol	Host Name	Server Name
sf_on	**onsoctcp**	sanfrancisco	sf_serv
oak_on	**onsocssl**	oakland	oak_serv

   For more information about the `sqlhosts` file and the SQLHOSTS registry, see the *IBM Informix Administrator's Guide*.

2. Using a text editor, create a `conssl.cfg` file in the `$INFORMIXDIR/etc` directory. The file must contain the following information:
   - **SSL_KEYSTORE_FILE** information that specifies the fully qualified file name of the keystore that stores the root CA certificates of all of the servers to which the client connects
   - **SSL_KEYSTORE_STH** information that specifies the fully qualified file name of the stash file containing the encrypted keystore password.

   The format of the `conssl.cfg` file is:

   ```
   Parameter    Value   # Comment
   ```

   For example, the `conssl.cfg` file might contain this information:

```
SSL_KEYSTORE_FILE   /work/keystores/ssl_client.kdb   # Keystore file
SSL_KEYSTORE_STH    /work/keystores/ssl_client.sth   # Keystore stash file
```

3. Use the GSKCapiCmd command-line interface, which is a part of JRE 1.6 or later, to set up a keystore and its password stash file and digital certificate.

   When you create the password, be sure that:

   - You use the command associated with the installed version of GSKit (for example, gsk7capicmd or gsk8capicmd).
   - The name and location of the keystore and its stash file are as specified in the `conssl.cfg` file.
   - Permissions on the keystore and its stash file are set to 666, even though the permissions are not enforced.

   If the certificate created for server is self-signed, you must extract the certificate from the server and use FTP to move the extracted certificate to the client, for the client keystore to use. If use the default certificates that are provided, you must create the client keystore.

   For example:

   - If the certificate is self-signed or is a default CA certificate, run the following commands on the client to create the keystore and add your certificate:

     ```
     GSK_COMMAND -keydb -create -db client.kdb -pw PASSWORD -type cms -stash
     ```

   - If the certificate created for the server is self-signed, additionally:

     a. Log on to the remote server and extract the certificate from the server keystore:

        ```
        GSK_COMMAND -cert -extract -db $INFORMIXSERVER.kdb -format ascii   -label
          SSL_KEYSTORE_LABEL -pw PASSWORD -target SSL_KEYSTORE_LABEL.cert
        ```

     b. Use FTP to move the extracted certificate to your client.

     c. Add the certificate to the client keystore:

        ```
        GSK_COMMAND -cert -add -db client.kdb -pw PASSWORD -label
          SSL_KEYSTORE_LABEL   -file SSL_KEYSTORE_LABEL.cert -format ascii
        ```

4. Add the digital certificate of the Certificate Authority that issued the server digital certificate to the keystore.

Related reference:

GSKIT_VERSION configuration parameter (Administrator's Reference)

Configuring server-to-server SSL connections

You can configure a high-availability data replication (HDR) primary server, an HDR secondary server, a shared disk (SD) secondary server, a remote stand-alone secondary (RSS) server, an Enterprise Replication node, or a server involved in a distributed transaction connection for Secure Sockets Layer (SSL) connections.

To configure HDR servers, Enterprise Replication nodes, or servers involved in a distributed transaction:

1. Configure each server for SSL connections. Follow the steps in "Configuring a server instance for secure sockets layer connections" on page 2-14.
2. In each server keystore, add the root digital certificate that the Certificate Authority (CA) issued to the other servers to the server keystore.

For example, suppose you have three servers: serv1 (the primary server), serv2 (the secondary server), and serv3 (a shared disk secondary server). Each server has its own keystore and digital certificate (serv1.kdb and serv1_label, serv2.kdb and serv2_label, serv3.kdb and serv3_label).

Add the root certificates that the Certificate Authority (CA) issued to each server to the other servers, as follows.

1. Add the root certificates issued to serv2 and serv3 to the serv1 keystore.
2. Add the root certificates issued to serv1 and serv3 to the serv2 keystore.
3. Add the root certificates issued to serv1 and serv2 to the serv3 keystore.

Related reference:

GSKIT_VERSION configuration parameter (Administrator's Reference)

Chapter 3. Column-level encryption

You can use column-level encryption to store sensitive data in an encrypted format. After encrypting sensitive data, such as credit card numbers, only users who can provide a secret password can decrypt the data.

Use the built-in ENCRYPT_AES() and ENCRYPT_TDES() encryption functions to encrypt data in columns containing the following character data types or smart large object data types:

- CHAR
- NCHAR
- VARCHAR
- NVARCHAR
- LVARCHAR
- BLOB
- CLOB

You can also use the SET ENCRYPTION PASSWORD statement to set an encryption password for a session. If you do this, only users who can provide a secret password can view, copy, or modify encrypted data.

The built-in ENCRYPT_AES(), ENCRYPT_TDES(), DECRYPT_CHAR(), and DECRYPT_BINARY() encryption and decryption functions can use the session-level password if the password is not explicitly specified in the encryption or decryption function. If you use the SET ENCRYPTION PASSWORD statement, you are not required to provide the same password in every encryption or decryption function.

After Informix prepares a statement that contains a password (and, optionally, a hint), Informix keeps the password and hint in shared memory in an encrypted format. Informix only decrypts a copy of the password or hint when any statement related to encryption is being executed. Informix uses a randomly generated session key to encrypt the password and hint in memory. This means that if the server fails with an AF (assertion failure) error, or if the shared memory is paged out of main memory, it is hard to find plain text passwords in the core dump. Informix never writes a password to disk. It records hints with the encrypted data, but they are lightly encrypted and not readily understood.

When you set encryption passwords for column data, you can specify these types of encryption:

- **Column-level encryption**. All values in a specific column of a database table are encrypted with the same password (word or phrase), the same encryption algorithm, and the same cipher mode. For column-level encryption, you can store the hint outside the encrypted column, rather than repeating it in every row.

  Tip: If encryption functions are not used, users can enter unencrypted data into columns that are meant to contain encrypted data. To ensure that data entered into a field is always encrypted, use views and INSTEAD OF triggers.

- **Cell-level encryption** (also called *row-column* or *set-column level encryption*). Within a column of encrypted data, many different passwords, encryption algorithms, or modes are used. This type of encryption might be necessary to protect personal data.

Passwords and hints that you declare with SET ENCRYPTION PASSWORD are not stored as plain text in any table of the system catalog. To prevent other users from accessing the plain text of encrypted data or of a password, you must avoid actions that might compromise the secrecy of a password:

- Unless your database is accessible only by a secure network, you must enable the Encryption Communication Support Module (ENCCSM) to protect data transmission between the database server and any client system.
- Do not index encrypted columns and do not create a functional index on a decrypted column. This would store plain-text data in the database, defeating the purpose of encryption.
- Do not store passwords in a trigger or in a user-defined routine (UDR) that exposes the password to the public. Use the session password before you activate the trigger, invoke the UDR, or pass any password as a parameter to a UDR.

When you set a password, the database server transfers the password and any hint to a 128-bit key that is used to encrypt the password and hint. Passwords and hints are not stored as clear text. The key is a time-based random value per instance. The database server starts the key when the server starts; the key is deleted when the database server shuts down.

Although it is possible to store both encrypted and unencrypted data in a single column, your application must determine which rows contain encrypted data and which rows contain unencrypted data. In addition, the application must provide for using the correct code to handle the difference, because the built-in decryption functions fail if they are applied to unencrypted data. The simplest way to avoid this error is for all rows to use encryption in a column where any row is encrypted. For more information, see the *IBM Informix Guide to SQL: Syntax*.

A query for encrypted data must specify an unencrypted column on which to select the rows. For information about queries, syntax, and reusing encrypted data, see the *IBM Informix Guide to SQL: Syntax*.

An encrypted value uses more storage space in a column than the corresponding plain text value. This occurs because all of the information required to decrypt the value, except the encryption key, is stored with the value. Therefore, embedding zero bytes in the encrypted result is not recommended.

The database server includes an Encrypt Virtual Processor. If the `encrypt` option of the **VPCLASS** parameter is not defined in the `onconfig` file, the database server starts one Encrypt VP the first time that any encryption or decryption functions defined for column-level encryption are called. You can define multiple Encrypt VPs if necessary to decrease the time required to start the database server. For more information, the configuration parameters chapter in the *IBM Informix Administrator's Reference*.

When the database server is in online mode, you can use the **onmode -p** command to add or drop Encrypt VPs. For example, to add four more Encrypt VPs, use:

```
onmode -p 4 encrypt
```

To drop three Encrypt VPs, use:

```
onmode -p -3 encrypt
```

For more information, see the **onmode** utility chapter in the *IBM Informix Administrator's Reference*.

Encrypting column data

You can store sensitive data in encrypted format.

Before you set the encryption password and encrypt data, you must be sure the encrypted data can fit in the column.

To encrypt a column:

1. Calculate the size of the encrypted column. If necessary, modify the column. For examples of two methods for calculating the size of an encrypted column, see "Example showing how to determine the size of an encrypted column."
2. Insert information about the encryption password into your code. Use the SET ENCRYPTION PASSWORD statement to specify either a password or a password and a hint. Use the ENCRYPT_AES() or the ENCRYPT_TDES() function to define encrypted data. For an example of how to insert a password into your code and use the ENCRYPT function, see "Example showing how to encrypt a column" on page 3-4.

Use the DECRYPT_BINARY(), and DECRYPT_CHAR() functions to query encrypted data. For an example of querying encrypted data, see "Example showing how to query encrypted data" on page 3-4.

See the *IBM Informix Guide to SQL: Syntax* for more information about:

- The SET ENCRYPTION PASSWORD statement and the syntax to use to specify the password and the hint
- The ENCRYPT and DECRYPT functions

Example showing how to determine the size of an encrypted column

The size of the column must be large enough to store the encrypted data.

The following example shows how the size of a Credit Card column is calculated:

```
DATA SIZE  16 bytes
    ENCRYPTED DATA SIZE = (DATA SIZE + blocksize8) / blocksize8 *
blocksize8 = 24 bytes (integer operation)
    OR ENCRYPTED DATA SIZE = (DATA SIZE - DATA SIZE% blocksize8 +
blocksize8 ) = 24 bytes
    (For ENCRYPT_TDES, round up to (N + 1) * 8 bytes, for example
13 bytes round up to 16 bytes, 16 bytes to 24 bytes)
    HEADER SIZE = 11 bytes (for Base64 encoding)
    IV SIZE = 8 bytes (fixed size)
    HINT SIZE = 32 bytes (maximum size)
    ENCRYPTED HINT SIZE = 40 bytes (maximum size)

BASE64 SIZE = ((INPUT DATA SIZE + 2) / 3) * 4
(integer operation)
    OR BASE64 SIZE = ((INPUT DATA SIZE + 2) -
(INPUT DATA SIZE + 2) % 3) / 3 * 4

    TOTAL SIZE = HEADER SIZE
```

```
    + BASE64(IV SIZE + ENCRYPTED DATA SIZE + ENCRYPTED HINT)
                         = 11 + BASE64(8 + 24 + 40)
                = 11 + (72 + 2) / 3 * 4
                = 11 + 96 = 107
```

In the previous example, Initialization Vector (IV) is a pseudo-random series of bytes that is used to initiate encryption when using some cipher modes. IV size is the number of random series of bytes; for Informix, this is 8 bytes.

If the hint is not stored in the column, the total size in the previous example is 55 bytes.

Another way to determine the encrypted column size is to calculate as follows:

```
SELECT LENGTH(ENCRYPT_TDES
("1234567890123456",
         "password", "long....hint"))
FROM "informix".systables WHERE tabid = 1
```

Without the hint, you can calculate as follows:

```
SELECT LENGTH(ENCRYPT_TDES("1234567890123456",
"password", ""))
FROM "informix".systables
WHERE tabid = 1
```

Important: If the column size is smaller than the returned data size from ENCRYPT and DECRYPT functions, the encrypted data is truncated when it is inserted and it is not possible to decrypt the data (because the header indicates that the length must be longer than the data received).

Example showing how to encrypt a column

You can use the SET ENCRYPTION PASSWORD statement to restrict access to data in a column.

The following example shows how to use the encryption password in a column that contains a social security number:

```
create table emp
(   name char(40),
    salary money,
    ssn lvarchar(67)
    );
        set encryption password "one two three 123";
        insert into emp values ("Alice", 50000, encrypt_aes
('123-456-7890'));
        insert into emp values ("Bob", 65000, encrypt_aes
('213-656-0890'));
        select name, salary, decrypt_char(ssn, "one two three 123")
 from emp where name = 'Bob';
```

Example showing how to query encrypted data

You can query encrypted data with the DECRYPT function or the SET ENCRYPTION PASSWORD statement.

The following example shows how to use the decrypt function to query encrypted data:

```
select name, decrypt_char(ssn, "one two three 123") from emp;
       or
set encryption password "one two three 123";
select name, salary, decrypt_char(ssn) from emp where name = 'Bob';
```

Chapter 4. Connection security

You can administer the security of the connections to the database server by using authentication and authorization processes.

The first step toward connecting to an Informix server is authentication. Authentication is the mechanism of verifying the identity of a user or an application. The Informix server supports a traditional authentication mechanism in which a user must provide a valid user ID and password combination to connect to a database. But you can configure the database server to add or modify an authentication mechanism. See "Authentication mechanisms" on page 4-2.

By default in standard installations, access to the database server also requires that the authentication credentials match the credentials of an OS user account on the Informix host computer. However, you can change the **USERMAPPING** parameter setting in the `onconfig` file to selectively remove the dependency on local OS user accounts and to enable a database server administrator (DBSA) to grant database server access to specific users without the OS user accounts. See "Internal users (UNIX, Linux)" on page 4-3.

With a non-root installation, the user who installs the product is the DBSA and typically chooses this type of installation because it requires less system administrator overhead than a standard installation. The database server of a non-root installation cannot authenticate users or applications based solely on their login to the local OS. The DBSA must set up internal users to grant database access to others. Typically, the number of database users with a non-root installation is lower than with many standard installations. See "Connections to a non-root installation (UNIX, Linux)" on page 4-8.

Authenticated users must specify a database to which to connect. A user can perform certain database actions or access certain database objects only if they have been authorized to do so by the DBA. For example, users with CONNECT privileges can connect to a database and run queries, while users with RESOURCE privileges can also create objects. See the *IBM Informix Guide to SQL: Syntax* for details about database-level privileges.

On a multitier network, you can create trusted connections between an application server and the Informix database server. You can use trusted connections to set the identity of each specific user accessing a database through the middle-tier server, which facilitates discretionary access control and auditing based on user identity. Without a trusted connection in such an environment, each action on a database is performed with the single user ID of the middle-tier server, potentially lessening granular control and oversight of database security. See "Trusted-context objects and trusted connections" on page 4-14.

You can ensure that connection authentication passwords are secure by encrypting them by using a communication support module (CSM). The simple password CSM (SPWDCSM) provides password encryption. SPWDCSM is available on all platforms. See "Simple password encryption" on page 4-31.

If you want to support a single sign-on (SSO) environment, you can use the Generic Security Services CSM (GSSCM) to implement a Kerberos authentication layer. In addition, the Kerberos protocol has several built-in features that can

provide the same security benefits that simple password CSM and encryption CSM have. SSO authentication verifies a user's identity, and it facilitates centralized management of user IDs and passwords. If confidentiality and integrity services are enabled in GSSCSM, Kerberos authentication encrypts data transmissions and ensures that transmissions are not altered between legitimate user and the database server.

Enterprise Replication and high availability connections cannot use authentication modules, but can function with these modules by restricting specific network ports to the replication and high availability connections. See Configuring secure connections for replication servers (Enterprise Replication) or Configuring secure connections for clusters (Administrator's Guide).

You can configure IBM Informix to check whether the ID of the user who is running the program matches the ID of the user who is trying to connect to the database. See "Securing local connections to a host" on page 4-41.

You can limit the ability of denial-of-service attacks to prevent legitimate connections to the database server from being blocked. See "Limiting denial-of-service flood attacks" on page 4-42.

Authentication mechanisms

You can configure the Informix server authentication mechanisms to meet varying requirements, such as different security methods required for local and remote connections, database access by users without operating system accounts on the servers host computer, and non-root installation.

Authentication is the mechanism of verifying the identity of a user or an application.

The simplest, default authentication method operates for a local connection by relying on OS user lookup. For this type of connection, a user ID and password pair are passed directly to the OS for verification that the user is legitimate. This method requires that users are granted connection privileges by the DBSA and have corresponding OS user accounts on the Informix host computer.

Starting with the Version 11.70 release on UNIX and Linux, an Informix installation can be configured to support other authentication mechanisms that maintain security while reducing the dependency on system administrator and root-level privileges.

Authentication layers

You can develop modules and configure a server to have a self-defined authentication mechanism for local and remote connections. An authentication-layer mechanism can function so that you are not required to make changes in the application. The database server supports these authentication layers:

- Pluggable Authentication Modules (PAM) for IBM Informix systems running on UNIX or Linux. The PAM framework provides a set of APIs for authentication, account, session, and password management.
- Lightweight Directory Access Protocol (LDAP) Authentication Support for Windows. Use the LDAP Authentication Support module when you want to use an LDAP server to authenticate users.

An Informix client can be a local or a remote user. For network-based business models, the database server uses the network authentication mechanism provided by the OS, but requires the DBSA to set up trusted-hosts information or trusted-user information. Trusted-hosts information is set in the `hosts.equiv` file or the file specified by the REMOTE_SERVER_CFG configuration parameter. Trusted-user information is set in each user's `rhosts` file or in the file specified by the REMOTE_USERS_CFG configuration parameter. You can modify lookup options in the `sqlhosts` file.

Users that connect to the database server without login to the host computer OS are *internal users*.

Related concepts:

"Internal users (UNIX, Linux)"

"Pluggable authentication modules for systems running on UNIX or Linux" on page 4-24

"LDAP authentication support on Windows" on page 4-26

Related tasks:

"Creating database server users (UNIX, Linux)" on page 4-9

"Maintaining database server users (UNIX, Linux)" on page 4-10

Related reference:

REMOTE_SERVER_CFG configuration parameter (Administrator's Reference)

REMOTE_USERS_CFG configuration parameter (Administrator's Reference)

sqlhosts file and SQLHOSTS registry key options (Administrator's Guide)

Internal users (UNIX, Linux)

The DBSA can grant database access to users that do not authenticate on the OS of the host computer by mapping PAM-authenticated users to OS-level entities or by configuring the server to perform internal authentication.

In Informix releases before Version 11.70, the database server was capable of identifying users through system calls that verified the credentials of the user attempting to connect through an OS account. Beginning with Version 11.70, internally authenticated users can connect even if the user cannot be identified by the OS.

Removing the dependency on a local host OS account for database server-access can reduce administrative work. For internal users, the DBSA is not required to coordinate with the OS Administrator to ensure that every user that must have server access also has an OS account.

There are two different types of internal users: mapped users and internally authenticated users. A mapped user connects to a database after providing a password that is validated in an authentication layer outside the database server. An internally authenticated user must provide a password matching one stored on the server, where authentication is an internal process.

Related concepts:

"Authentication mechanisms" on page 4-2

Mapped users (UNIX, Linux)

The DBSA can configure the server to allow database access by external users. External users attempting to connect through Kerberos single sign-on (SSO), a Pluggable Authentication Module (PAM), or internal authentication can be mapped to an OS-level profile for processing connection requests.

Sometimes running an SQL statement requires the database server to interact with the OS, typically to read or write a file, or to run a program through the SPL SYSTEM statement. When interaction with the OS is required, the database server must be provided OS credentials to manage the file or run the program.

Users can be mapped to one of the following surrogate user identities:

- A UID and GID pair defined in the database server
- An existing OS user account on the database server host computer

After a user authenticates, whenever the database server interacts with the OS on behalf of the user, the surrogate user properties specified by the user mapping are invoked. The simplest mapping is identity mapping when the user name maps directly to the OS properties of a user with the same name. If you are the OS Administrator, you can use the `/etc/informix/allowed.surrogates` file to specify which surrogate users and groups can be used so that mapped users are not granted owner access to sensitive systems, such as databases, print spoolers, email, or the operating system, itself.

Note: The `allowed.surrogates` file is not used or read by non-root installations of the database server, because the database server does not perform operations or run commands as the user who started the session.

The CREATE USER and GRANT ACCESS TO PROPERTIES statements can create complex mappings of surrogate properties, including:

- user ID
- user name
- surrogate groups
- home directory
- authorization privilege (DBSA, DBSSO, AAO, or BARGROUP)

The CREATE USER and ALTER USER statements associate OS-level privileges by mapping users to OS properties and storing this information in a series of system catalog tables.

Note: In the 11.70.xC1 release, the only SQL statement that supported user mapping was GRANT ACCESS TO for users connecting through PAM or SSO. The GRANT ACCESS TO PROPERTIES statement is still compatible with mapped users created with the 11.70.xC1 release, but this SQL statement is deprecated for newer implementations of mapped users.

Users can be mapped DB-Access and the IBM OpenAdmin Tool (OAT) for Informix GUI. After a DBSA sets the USERMAPPING configuration parameter in the `onconfig` file, and maps externally authenticated users to surrogate properties

in tables of the SYSUSER database, it is possible for the mapped users to connect to the database server without a local OS account.

In order to enable mapped users functionality, the USERMAPPING configuration parameter must be set to either BASIC or ADMIN.

After you set the USERMAPPING configuration parameter to BASIC or ADMIN, you can use the following DDL operations on mapped users:

- ALTER USER
- CREATE USER
- DROP USER
- GRANT ACCESS TO PROPERTIES
- SET USER PASSWORD
- RENAME USER
- REVOKE ACCESS

Related concepts:

"Pluggable authentication modules for systems running on UNIX or Linux" on page 4-24

Related tasks:

"Setting up an SSO authentication environment" on page 4-35

Related reference:

USERMAPPING configuration parameter (UNIX, Linux) (Administrator's Reference)

Surrogate user properties (UNIX, Linux) (SQL Syntax)

Mapped user surrogates in the allowed.surrogates file (UNIX, Linux)

The OS Administrator can use the `/etc/informix/allowed.surrogates` file to control which OS users and groups can act as surrogates for mapped users.

The database server uses surrogate user properties while it performs operating system operations on behalf of a mapped user. OS user names, user IDs, group names, and group IDs specified in `/etc/informix/allowed.surrogates` file are cached in shared memory during server start-up and after you run **onmode -cache surrogates**, and are checked during user creation and before the user is allowed to connect to the server.

The **onmode -cache surrogates** command causes the server to reread the `allowed.surrogates` file and store the user names, user IDs, group names, and group IDs values in shared memory cache. If the cache-refresh fails, previously stored surrogate names are cleared from the cache, effectively disabling mapped users. Changes in shared memory cache affect new sessions. Existing connections on the server are unaffected.

The improved control makes root installations of Informix more secure by preventing the DBSA from specifying surrogates that could compromise operating system security.

Note: The `allowed.surrogates` file is not used or read by non-root installations of Informix, because the database server does not perform operations or run commands as the user who started the session.

Related reference:

onmode -cache surrogates: Cache the allowed.surrogates file (Administrator's Reference)

CREATE USER statement (UNIX, Linux) (SQL Syntax)

Surrogate user properties (UNIX, Linux) (SQL Syntax)

Specifying surrogates for mapped users (UNIX, Linux)

Specify operating system (OS) user names, user IDs, group names, and group IDs in the `allowed.surrogates` file to control which OS users and groups can act as surrogates for mapped users.

1. Create a file named `allowed.surrogates` in the `/etc/informix` directory. The `allowed.surrogates` file must be owned by **root** instead of **informix**. The file must not have execute permissions and only the file owner can have write permission.
2. In the `allowed.surrogates` file, enter the OS user names, user IDs, OS group names, group IDs, ranges of user IDs, and ranges of group IDs that you want to allow as surrogates.
   a. Enter comma-separated OS user names, user IDs, and ranges of user IDs after entering the `user:` label.

      ```
      users:user1,user2,105,104,300,400..500
      ```

   b. Enter comma-separated OS group names, group IDs, and ranges of group IDs after entering the `group:` label.

      ```
      groups:ifx_dbsa,group1,group2,root,1,10..20
      ```

   The group and user labels are case-insensitive, and can be pluralized. Entries are separated by commas. Ranges of user IDs and group IDs are inclusive, with the upper and lower ranges separated by two periods. You must specify both an upper and lower limit for ranges. Comment lines begin with # and are ignored. Blank lines are also ignored.

   If the `allowed.surrogates` file is formatted incorrectly, then user mapping is disabled and an error is logged in the online log file. If a user name or group name cannot be identified, the name is logged in the online log file and otherwise ignored, and the cache is cleared.

The following example of an `allowed.surrogates` file entry specifies user **user1**, **user 40**, **users 45-50**, and **group 10** as acceptable surrogates.

```
#Surrogate IDs
USERS:user1,40,45..50
GROUP:10
```

Related reference:

onmode -cache surrogates: Cache the allowed.surrogates file (Administrator's Reference)

CREATE USER statement (UNIX, Linux) (SQL Syntax)

Surrogate user properties (UNIX, Linux) (SQL Syntax)

Internally authenticated users (UNIX, Linux)

The DBSA can configure the server to authenticate users by checking their credentials with a hashed password that is stored inside the database server.

The DBSA creates internally authenticated users with the CREATE USER statement and sets up a password that is stored in the Informix SYSINTAUTHUSERS catalog table of the SYSUSER database.

The DBSA can administer internally authenticated users with the CREATE USER, DROP USER, ALTER USER, and RENAME USER SQL statements. Users can change their own password with the SET USER PASSWORD statement.

Related concepts:

"User mapping tables (UNIX, Linux)" on page 4-11

Related reference:

ALTER USER statement (UNIX, Linux) (SQL Syntax)

DROP USER statement (UNIX, Linux) (SQL Syntax)

RENAME USER statement (UNIX, Linux) (SQL Syntax)

SET USER PASSWORD statement (UNIX, Linux) (SQL Syntax)

Location and file names for mapped users' generated files (UNIX, Linux)

The generated files for some mapped users are created in a different default location, and are named differently.

If a mapped user is created with a non-OS surrogate UID and a home directory is not specified for this user, the working directory of the user is set to `$INFORMIXDIR/users/server_servernumber/uid_UID`. Generated files, such as explain output files and debug files are stored in the user's working directory.

Generated files are renamed to have a *username_* prefix, where *username* is the name of mapped user. For example on server number **2**, the `sqexplain.out` file for mapped user **fred**, who has a UID of **3000**, would be `$INFORMIXDIR/users/server_2/uid_3000/fred_sqexplain.out`

If the length of the directory exceeds 256 bytes, the database server generates a warning, and files are generated in `$INFORMIXDIR`, instead.

If a CREATE USER or ALTER USER statement was used to specify the mapped user's home directory, then remote clients' files are generated in user's home directory and local clients' files are generated in the user's current working directory. File names changed to have a *username_* prefix, where *username* is the name of the mapped user. Distinguishing the files of a specific mapped user is important, because multiple users can be mapped to the same surrogate UID.

If there are characters in a mapped user's name that interfere with file creation, characters are changed to '_'. For example, the generated files of mapped user **foo/bar** are renamed with the prefix `foo_bar_`.

Related concepts:

Default name and location of the explain output file on UNIX (SQL Syntax)

Related reference:

CREATE USER statement (UNIX, Linux) (SQL Syntax)

ALTER USER statement (UNIX, Linux) (SQL Syntax)

Surrogate user properties (UNIX, Linux) (SQL Syntax)

USERMAPPING configuration parameter (UNIX, Linux) (Administrator's Reference)

Connections to a non-root installation (UNIX, Linux)

A non-root installation of a database server runs without user **root** or user **informix** privileges. Installation and user administration of a non-root server can be easier, but this type of installation does not support all product features.

A user who installs the database server without user **root** privileges performs a non-root installation. The user who completes non-root installation is the DBSA and creates and controls the other user accounts that can connect to the server. Installation and administration of the server does not depend on the root-level privileges associated with user **informix** and group **informix**.

The non-root server cannot authenticate users based on system calls to the OS level. The DBSA of the installation must create internal users to grant database server access to other users.

Consider non-root installation for easier product setup and embeddability, or for security in an environment that does not require high scalability. This type of installation might be preferable in environments where access to root privileges is rigorously controlled and it is difficult to obtain permission to create accounts like user **informix** and group **informix**.

Important: See "Non-root installation" in the *IBM Informix Installation Guide for UNIX, Linux, and Mac OS X* for important information about product features not supported with a server running without root privileges.

Shared-memory and stream-pipe connections to a non-root installation

You must include the `cfd` option in the `sqlhosts` file to use shared-memory and stream-pipe connections on servers that have a non-root installation of Informix.

For root installations of the database server, communication files required for shared-memory and stream-pipe connections are stored in and accessed from the `/INFORMIXTMP` directory, a location where a non-root installation of Informix does not have write permission.

Use the `cfd` option in the `sqlhosts` file to set the directory where you want to store the communication files necessary for shared-memory and stream-pipe connections. Non-root installations of Informix now store communication files in the `$INFORMIXDIR/etc`directory by default. Clients connecting to the server now check the `$INFORMIX/etc` directory if communication files are not found in the `/INFORMIXTMP` directory.

Non-root installations of the database server do not have permission to write to the /INFORMIXTMP directory, so shared-memory and stream-pipe connection communication files are not written to the $INFORMIXDIR/etc directory if no communication files directory is specified as an option in the sqlhosts file.

Clients connecting to an Informix server first check the /INFORMIXTMP directory for communication files before checking the $INFORMIXDIR/etc directory for communication files. If the client and server for a non-root installation of Informix are located in different locations, then the communication files are not in the /INFORMIXTMP or $INFORMIXDIR/etc directories, and the connection fails. In this case, you must specify the cfd value in the sqlhosts file for the connection attempt to succeed.

Related concepts:

Non-root installation (Installation Guide (UNIX))

Related tasks:

"Creating database server users (UNIX, Linux)"

Creating database server users (UNIX, Linux)

If you have DBSA privileges, you can create internally authenticated users or you can create users who do not have accounts on the host system. To create these types of users, you must map each user to the appropriate user and group privileges, regardless of whether these users have operating system accounts on the database server host computer.

After a non-root database server is installed, users cannot immediately connect to the server with passwords because permission issues prevent OS authentication. Additionally, users do not yet exist in the internal database. The only way to initially connect to a non-root server is without a password. Because only a DBSA can create users, the database owner must make a connection without a password, and then create users in the database. The DBSA can create a user with or without a password. The method of establishing the initial connection without a password is provided in this task.

Prerequisites:

- You must have DBSA privileges. By default, the owner of private server is a DBSA. When you create or modify user accounts, you can use CREATE USER or ALTER USER statements to grant the DBSA privilege to other users.
- For a non-root installation only: After installation, you must connect to the database server by using DB-Access.

  On local clients, you can start DB-Access and establish a connection to the server by using a user name and password. Alternatively, on the command prompt, a user can run the **dbaccess** command and then run other SQL statements to connect without a password, as follows:

  ```
  >dbaccess - -
  > database  sysuser;

  Database selected.

  >
  ```

  If you want to connect from a remote computer without a password, you must have trusted-host information or trusted-user information specified. Trusted-host information is in the hosts.equiv file or the file specified by the

REMOTE_SERVER_CFG configuration parameter. Trusted-user information is in each user's `rhosts` file or the file specified by the REMOTE_USERS_CFG configuration parameter.

To create user accounts for database users:

Run the CREATE USER statement, in the format shown in the following examples:

```
CREATE DEFAULT USER WITH PROPERTIES USER 'guest';
```

```
CREATE USER username WITH PASSWORD password
```

Enabling a new user to successfully connect to the server:

You are not required to specify information in the USERMAPPING configuration parameter when you create users. However, if you want to enable the mapped or internal user to successfully connect to the server, you must set the USERMAPPING configuration parameter, as follows:

- If you do not want mapped users to have administrative privileges, set the USERMAPPING parameter to `BASIC`.
- If you want to make it possible for selected mapped users to have administrative privileges, set the USERMAPPING parameter to `ADMIN`.

  No administrative privileges are given to any users until you provide that access when you run a CREATE USER (or ALTER USER) statement. You can grant ADMIN privileges to users with surrogate property AUTHORIZATION. The valid values are `dbsa`, `dbsso`, `aao` and `bargroup`.

Related concepts:

"User mapping tables (UNIX, Linux)" on page 4-11

"Connections to a non-root installation (UNIX, Linux)" on page 4-8

"Authentication mechanisms" on page 4-2

Related tasks:

"Maintaining database server users (UNIX, Linux)"

Related reference:

USERMAPPING configuration parameter (UNIX, Linux) (Administrator's Reference)

Connect to a database environment (DB-Access Guide)

CREATE USER statement (UNIX, Linux) (SQL Syntax)

Maintaining database server users (UNIX, Linux)

You can modify, drop, or rename a database server user account that was created with the CREATE USER statement.

Prerequisites:

- You must have DBSA privileges.

To manage user accounts:

Run one of the following statements, as appropriate:

- Run the ALTER USER statement to change user properties such as user, password, groups, authorization privilege, home directory, and to enable or disable the account of an internally authenticated user, or of the default internally authenticated user.

- Run the DROP USER statement to remove the user.
- Run the RENAME USER statement to give the user a different name.

If you are a user whose account was created with the CREATE USER statement, you can run the SET USER PASSWORD statement to change your password.

Examples:

To grant DBSA authorization to a user named **foo**, run the following statement:

```
ALTER USER foo ADD AUTHORIZATION (dbsa);
```

To remove the DBSA authorization from user **foo**, run the following statement:

```
ALTER USER foo DROP AUTHORIZATION (dbsa);
```

To remove a user named **david**, run this command:

```
DROP USER david
```

To give a user named **ann** the new name of ann2, run this command:

```
RENAME USER ann to ann2
```

To change a password to **pwd3439** for a user named **radha**, run this command

```
ALTER USER radha MODIFY PASSWORD pwd3439
```

If you are a user, not a DBSA, you can change your password, for example, from **js12342394** to **jayal89**, by running this command:

```
SET USER PASSWORD OLD js12342394 NEW jayal89
```

Related concepts:

"Authentication mechanisms" on page 4-2

"User mapping tables (UNIX, Linux)"

Related tasks:

"Creating database server users (UNIX, Linux)" on page 4-9

Related reference:

ALTER USER statement (UNIX, Linux) (SQL Syntax)

DROP USER statement (UNIX, Linux) (SQL Syntax)

RENAME USER statement (UNIX, Linux) (SQL Syntax)

SET USER PASSWORD statement (UNIX, Linux) (SQL Syntax)

User mapping tables (UNIX, Linux)

The user mapping tables in the SYSUSER database are system tables that map users to OS-level properties that enable IBM Informix access and control level of discretionary access privileges.

sysusermap table

Database: SYSUSER

Table 4-1. Schema of the sysusermap table

Column	Type	Description
username	CHAR(32)	PUBLIC or a mapped user name

Table 4-1. Schema of the sysusermap table (continued)

Column	Type	Description
surrogate_id	INT	Identification number for a surrogate user identity. This number is generated when you run the GRANT ACCESS TO statement to create a mapped user.

syssurrogates table

Database: SYSUSER

Table 4-2. Schema of the syssurrogates table

Column	Type	Description
surrogate_ID	SERIAL	Identification number for a surrogate user identity. This number is generated when you run the GRANT ACCESS TO statement to create a mapped user.
os_username	CHAR(32)	User name of an operating system account on the IBM Informix host computer to be used as the surrogate user identity. The os_username field is null when you set a value to the UID keyword in the GRANT ACCESS TO statement.
uid	INT	User identifier number that corresponds with the permissions to which you want to map a user, users, or PUBLIC. This number and the corresponding *gid* value together form a surrogate user identity. The uid field is null when you specify a name with USER keyword in the GRANT ACCESS TO statement.
gid	INT	Group identifier number that corresponds with the permissions to which you want to map a user, users, or PUBLIC.
groupname	CHAR(32)	A group name that exists on the operating system of the IBM Informix host computer.

Table 4-2. Schema of the syssurrogates table (continued)

Column	Type	Description
homedir	VARCHAR(255)	Full path name in which user files are stored. The uid and gid must own the directory and have READ, WRITE, and EXECUTE permissions. The directory must not have PUBLIC WRITE permission.
userauth	CHAR(10)	Contains userauth pattern that indicates whether the user has server administrator privileges.

syssurrogategroups table

Database: SYSUSER

Table 4-3. Schema of the syssurrogategroups table

Column	Type	Description
surrogate_id	INT	Identification number for a surrogate user identity. This number is generated when you run the GRANT ACCESS TO statement to create a mapped user.
gid	INT	Group identifier number that corresponds with the permissions to which you want to map a user, users, or PUBLIC.
groupname	CHAR(32)	A group name that exists on the operating system of the IBM Informix host computer.
groupseq	SMALLINT	Unique number associated with the group information.

sysintauthusers table

Database: SYSUSER

Before a user connects to a database of a non-root installation, the server must authenticate the user by verifying credentials in sysintauthusers table. The value that is stored in the sysintauthusers table of the sysusers database is hashed with a 64-bit random salt that is also stored.

Table 4-4. Schema of the sysintauthusers table

Column	Type	Description
username	NCHAR(32)	Name for the user.

Table 4-4. Schema of the sysintauthusers table (continued)

Column	Type	Description
salt	BIGINT	64-bit salt that the server uses to morph the password before applying the hashing algorithm. The server can use salt to change a password so that two users with the same password do not have the same hashed password in the database. Salt improves security because it prevents password guessing.
hashed_password	VARCHAR(128)	A sha-256 hashed and base-64 encoded password.
hash_type	CHAR(16)	Type of hashing algorithm used. Currently the SHA-256 algorithm is used.
updated	DATETIME YEAR TO SECOND {TIMESTAMP}	N/A
flags	INTEGER	Flags used to store some account information (such as the account lock).
min_change	INTERVAL DAY(7) TO SECOND	N/A
max_change	INTERVAL DAY(7) TO SECOND,	N/A
inactive	INTERVAL DAY(7) TO SECOND	N/A
ac_expire	DATETIME YEAR TO SECOND	N/A

Related concepts:

"Internally authenticated users (UNIX, Linux)" on page 4-6

Related tasks:

"Creating database server users (UNIX, Linux)" on page 4-9

"Maintaining database server users (UNIX, Linux)" on page 4-10

Related reference:

USERMAPPING configuration parameter (UNIX, Linux) (Administrator's Reference)

Trusted-context objects and trusted connections

You can use trusted-context objects and trusted connections to increase system performance and security within a three-tier application model.

Trusted connections are established through a trusted-context database object, which must be created and defined by a user who holds the DBSECADM role. Trusted-context objects can contain:

- Attributes for defining a trusted connection
- Authentication requirements for trusted-connection users

- Roles for defining the access privileges of trusted-connection users

If a trusted-connection request matches all of a trusted-context object's attributes, the system grants a trusted connection. If a connection request contains an attribute that does not match the trusted-context object, the system rejects the request.

If you are a Database Administrator, and users are accessing your database through a middle-tier server, you can use trusted-context objects to:

- Increase system security
- Increase overall system performance
- Reduce maintenance overhead
- Control user privileges
- Preserve auditing capability of user access

If you are an Application Developer, and you are accessing a database through a middle-tier server, you can use trusted connections to:

- Maintain your user ID when you access a database server
- Share a single database connection with other users
- Increases overall system performance

Related concepts:

"Database Security Administrator Role" on page 6-3

Security Administration Options (SQL Syntax)

Related reference:

CREATE TRUSTED CONTEXT statement (SQL Syntax)

The three-tier application model

The traditional three-tier application model has system performance and security issues that can be addressed by trusted-context objects and trusted connections.

The traditional three-tier application model

In a traditional three-tier application model, all interaction between users and a database server occurs through a database connection established in a middle-tier server. Users log into the middle tier, and the middle tier then logs into the database server. The ID and password stored on the middle tier, rather than the user's ID and password, are used for all authorization checking and auditing required to access the database server. All activity occurring through the middle tier is performed and recorded as if it comes from a single user, rather than from the multiple users that log into the middle tier.

Security drawbacks of the traditional three-tier application model

The loss of user identity that occurs on the traditional three-tier application model causes the following security problems:

- Diminished user accountability
- Inability for users with the DBSECADM role to set specific user access privileges, resulting in over-granting or over-restricting user access to resources

Connection options and related issues for some middle-tier servers

Some middle-tier servers are capable of establishing a new, different connection for each user ID and password. In this case, connection requests require that all user IDs and passwords be stored and authenticated in two places. The extra storage and connection requirements cause the following problems:

- Reduced system performance because creating new connections requires more system overhead
- Increased maintenance requirements for management of user IDs and passwords in multiple locations

Requirements for trusted-context objects and trusted connections

Before you create trusted-context objects and use trusted connections, you must ensure that system and user requirements are met.

To create trusted-context objects you must have a user ID that has been granted the DBSECADM role.

To create trusted connections, you must be using an application that connects to the Informix Database server through TCP/IP. Local communication protocols are not supported. The following APIs can be used to request trusted connections:

- IBM Informix ESQL/C
- IBM Informix JDBC Driver
- IBM Informix ODBC Driver
- IBM Data Server Driver for JDBC and SQLJ
- IBM Data Server Provider for .NET

Related concepts:

"Database Security Administrator Role" on page 6-3

Related reference:

CREATE TRUSTED CONTEXT statement (SQL Syntax)

TCP/IP connectivity files (Administrator's Guide)

Creating a trusted-context object

You must create trusted-context objects before you can create trusted connections to a database server.

If you are managing trusted-connection users' access privileges, verify that the privileges available through currently defined ROLE objects are appropriate, or request that the Database Administrator define roles with privileges appropriate for users.

To create trusted-context objects, use the CREATE TRUSTED CONTEXT statement. Define the attributes of the object to meet the requirements of database users.

- After the CREATE TRUSTED CONTEXT clause, specify the name of the trusted-context object.
- After the `USER` keyword, specify the system authorization ID (user ID) of the primary user.

Note: The BASED UPON CONNECTION USING SYSTEM AUTHID clause used for DB2® servers also works in place of the USER keyword.

- After the ADDRESS keyword, specify the IPv4 addresses, IPv6 addresses, or secure domain names of all workstations that must use a trusted connection.

  Note: Locations based on Dynamic Host Configuration Protocol (DHCP) must not be used. Recycling IP addresses can result in unapproved users receiving trusted-locations status.
- Enter the ENABLE attribute to make the trusted-context object functional. Trusted-context objects have default state of DISABLE.
- If the connection is used by multiple, specific users, specify other trusted-connection users' IDs after the WITH USE FOR clause.
- If the connection is available to any user, enter the PUBLIC attribute after the WITH USE FOR clause.
- If you are specifying authentication (password) requirements for users, use the WITH AUTHENTICATION or WITHOUT AUTHENTICATION attributes after each user's ID or after the WITH USE FOR PUBLIC clause.
- If you are assigning roles to specific users, use ROLE keyword, followed by the role name, after the user's WITH AUTHENTICATION or WITHOUT AUTHENTICATION attributes.
- If you are assigning a default role to users, use the DEFAULT ROLE clause, followed by the role name. Trusted-context objects have default state of NO DEFAULT ROLE.

After you have created a trusted-context object, you can make changes to it by using the following statements:

- Use the ALTER TRUSTED CONTEXT statement to change the definition of a trusted-context object.
- Use the RENAME TRUSTED CONTEXT statement to change the name of a trusted-context object.
- Use the DROP TRUSTED CONTEXT statement to remove the trusted-context definition from the Informix system catalog.

Related concepts:

"Database Security Administrator Role" on page 6-3

Related reference:

CREATE TRUSTED CONTEXT statement (SQL Syntax)

ALTER TRUSTED CONTEXT statement (SQL Syntax)

DROP TRUSTED CONTEXT statement (SQL Syntax)

RENAME TRUSTED CONTEXT statement (SQL Syntax)

Examples of defining trusted locations

These trusted-context examples show how to define trusted locations by using the ADDRESS attribute.

Example 1: Using an IPv4 address

In this example, trusted-context object tcx1 grants user newton a trusted connection if the request comes from the IPv4 address 192.0.2.1.

```
CREATE TRUSTED CONTEXT tcx1
  USER newton
  ATTRIBUTES (ADDRESS '192.0.2.1')
  ENABLE
```

Example 2: Using an IPv6 address

In this example, trusted-context object tcx2 grants user brock a trusted connection if the request comes from the IPv6 address 2001:0DB8:0000:0000:0008:0800:200C:417A.

```
CREATE TRUSTED CONTEXT tcx2
  USER brock
  ATTRIBUTES (ADDRESS '2001:0DB8:0000:0000:0008:0800:200C:417A')
  ENABLE
```

Tip: You can use the compressed form of the IPv6 address: 2001:DB8::8:800:200C:417A.

Example 3: Using a secure domain name

In this example, trusted-context object tcx3 grants user hayes a trusted connection if the request comes from the secure corona.testlab.ibm.com domain.

```
CREATE TRUSTED CONTEXT tcx3
  USER hayes
  ATTRIBUTES (ADDRESS 'corona.testlab.ibm.com')
  ENABLE
```

Example 4: Defining multiple trusted locations in a single trusted-context object

In this example, trusted-context object tcx4 grants user newton a trusted connection if the request comes from the IPv4 address 192.0.2.1, the IPv4 address 194.0.6.3, or the secure corona.testlab.ibm.com domain.

```
CREATE TRUSTED CONTEXT tcx4
  USER newton
  ATTRIBUTES (ADDRESS '192.0.2.1',
              ADDRESS '194.0.6.3',
              ADDRESS 'corona.testlab.ibm.com')
  ENABLE
```

Related reference:

CREATE TRUSTED CONTEXT statement (SQL Syntax)

Examples of specifying authentication requirements for trusted connections

These examples show how to specify authentication requirements for trusted connections by using the WITH USE FOR clause and the WITH AUTHENTICATION and WITHOUT AUTHENTICATION attributes.

Example 1: Specifying authentication requirements for a group of users

The WITH USE FOR clause specifies which users can share a trusted connection. The PUBLIC attribute specifies that any user can connect on a trusted-connection switch request.

In this example, trusted-context object tcx1 grants user newton a trusted connection if the request is coming from the IPv4 address 192.0.2.1. The trusted connection can be switched to any other user, and switching does not require a password from the new user.

```
CREATE TRUSTED CONTEXT tcx1
  USER newton
  ATTRIBUTES (ADDRESS '192.0.2.1')
  ENABLE
  WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
```

Example 2: Specifying authentication requirements for specific users

The WITH AUTHENTICATION clause specifies that switch requests from the specified user or group of users require authentication (a password). WITH USE FOR clauses that do not specify WITH AUTHENTICATION do not require a password for switching.

In this example, the trusted-context object tcx2 grants user newton a trusted connection if the request is coming from the IPv4 address 192.0.2.1. The trusted connection can be switched to brock if brock provides a password. The trusted connection can be switched to hayes without a password.

```
CREATE TRUSTED CONTEXT tcx2
  USER newton
  ATTRIBUTES (ADDRESS '192.0.2.1')
  ENABLE
  WITH USE FOR brock WITH AUTHENTICATION,
               hayes WITHOUT AUTHENTICATION
```

Related reference:

CREATE TRUSTED CONTEXT statement (SQL Syntax)

Example of assigning a default role in a trusted-context object

This example demonstrates how to assign a default role for users of a trusted connection by using the DEFAULT ROLE clause. You can use the structure of this example to specify privileges for users of a trusted-context object.

Roles and privileges established through the trusted-context object allow a user to gain privileges in addition to the ones they already have.

A new user of a switched trusted connection inherits either a default role or a specific role from a trusted-context object. If a trusted-context object does not define a specific role for a trusted-connection user, the user inherits the default role, and all the access privileges that the Database Administrator defined for that default role.

In this example, the trusted-context object tcx1 grants user brock a trusted connection if the request is coming from the IPv4 address 192.0.2.1. The trusted connection that brock is granted can be switched to any user. brock and all other trusted connection users are granted the default MANAGER role, and all MANAGER privileges that were previously defined by the Database Administrator.

```
CREATE TRUSTED CONTEXT tcx1
  USER brock
  ATTRIBUTES (ADDRESS '192.0.2.1')
  DEFAULT ROLE MANAGER
  ENABLE
  WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
```

Related concepts:

Roles (Database Design and Implementation Guide)

"User roles" on page 5-1

"Default roles" on page 5-2

Related reference:

CREATE TRUSTED CONTEXT statement (SQL Syntax)

Example of assigning user-specific privileges in a trusted-context object

This example demonstrates how to assign user-specific privileges for a trusted connection by using the ROLE object. You can use the structure of this example to assign privileges for users of a trusted-context object.

In this example, the trusted-context object `tcx1` grants user `newton` a trusted connection if the request is coming from the IPv4 address `192.0.2.1`. The trusted connection that `newton` is granted can be switched to `brock` without a password. The trusted connection can be switched to `hayes`, but `hayes` must provide a password.

`newton` is granted the default `AUDITOR` role and privileges. If the connection is switched to `brock`, `brock` is granted the default `AUDITOR` role and privileges. If the connection is switched to `hayes`, `hayes` is granted the specific `MANAGER` role and privileges instead of the `AUDITOR` role and privileges.

```
CREATE TRUSTED CONTEXT tcx1
  USER newton
  ATTRIBUTES (ADDRESS '192.0.2.1')
  DEFAULT ROLE AUDITOR
  ENABLE
  WITH USE FOR brock WITHOUT AUTHENTICATION,
               hayes WITH AUTHENTICATION ROLE MANAGER
```

Related concepts:

Roles (Database Design and Implementation Guide)

"User roles" on page 5-1

Related reference:

CREATE TRUSTED CONTEXT statement (SQL Syntax)

Creating a trusted connection

Using your application, you can create a trusted connection to your database.

Before you can create a trusted connection, ensure that:

- The trusted-context object is enabled.
- Your user ID has CONNECT privileges to the database.
- Your user ID matches the primary user ID in the trusted-context object.
- Your connection request is coming from a trusted location that is defined in the trusted-context object.

To request a trusted connection, use the appropriate command from within your application.

- IBM Informix ESQL/C

  Use the `TRUSTED` keyword within the existing CONNECT statement.

  ```
  EXEC SQL CONNECT TO 'database_name' TRUSTED;
  ```

- IBM Informix JDBC Driver

  Include the TRUSTED_CONTEXT = TRUE; property in the database URL.

  ```
  jdbc:informix-sqli://hostname:portnumber/database_name:
        INFORMIXSERVER = server_name; TRUSTED_CONTEXT = TRUE;
  ```

- IBM Informix ODBC Driver

  Local transactions are supported on the IBM Informix ODBC Driver. but distributed (XA) transactions are not.

  For local transactions, call the SQLSetConnectAttr function to set the SQL_ATTR_USE_TRUSTED_CONTEXT attribute before you open a connection.

  ```
  SQLSetConnectAttr( hdbc, SQL_ATTR_USE_TRUSTED_CONTEXT,
        SQL_TRUE, SQL_IS_INTEGER );
  ```

  You can also request a connection by including the TCTX = 1 attribute in the connection string.

  ```
  SQLDriverConnect( hdbc, NULL, "DSN = MyDSN; TCTX = 1",
        SQL_NTS,ConnStrOutp, 250, &pcbConnStrOut, SQL_DRIVER_NOPROMPT );
  ```

  Note:

- IBM Data Server Driver for JDBC and SQLJ
  - For local transactions, use the getDB2TrustedPooledConnection method.

    ```
    getDB2TrustedPooledConnection( String user_ID, String user_password,
          java.util.Properties properties );
    ```

  - For distributed transactions, use one of the following getDB2TrustedXAConnection methods.

    ```
    getDB2TrustedXAConnection( String user_ID,
          String user_password, java.util.Properties properties );

    getDB2TrustedXAConnection( java.util.Properties properties );
    ```

- IBM Data Server Provider for .NET

  Use the TrustedContextSystemUserID and TrustedContextSystemPassword properties in the connection string.

  ```
  String connectString = "
     Server = IP_address/Local_host:port_number;
     Database = database_name;
     TrustedContextSystemUserID = user_ID;
     TrustedContextSystemPassword = user_password;
  ";
  ```

Related reference:

CONNECT statement (SQL Syntax)

Switching the user ID on a trusted connection

You can switch user IDs after a trusted connection is established.

Ensure that at least one of the following statements is true:

- The new user ID is the primary user ID defined in the trusted-context object.
- The new user ID is explicitly defined as a user in the trusted-context object.
- The trusted-context object is defined as WITH USE FOR PUBLIC.

Attention: During a user-ID switch, the database connection is maintained, but the switch results in a new connection environment. Objects such as temporary tables and WITH HOLD cursors are lost in the new environment.

To switch to a different user, use the appropriate command in your application:

- IBM Informix ESQL/C, IBM Informix JDBC Driver, and IBM Informix ODBC Driver
  - For a switch request without authentication requirements, use the SET SESSION AUTHORIZATION statement without a user password.

    ```
    SET SESSION AUTHORIZATION TO 'user_ID';
    ```

  - For a switch request with authentication requirements, include the new user's password in the SET SESSION AUTHORIZATION statement.

    ```
    SET SESSION AUTHORIZATION TO 'user_ID' USING 'user_password';
    ```

- **IBM Data Server Driver for JDBC and SQLJ**
  - For a switch request without authentication requirements on a local-transaction, trusted connection, use the getDB2TrustedPooledConnection object.

    ```
    getDB2TrustedPooledConnection( String user_ID,
          java.util.Properties properties );
    ```

  - For a switch request without authentication requirements on a distributed-transaction, trusted connection, use the getDB2TrustedXAConnection object.

    ```
    getDB2TrustedXAConnection(String user_ID,
          java.util.Properties properties );
    ```

  - For a switch request with authentication requirements on a local-transaction, trusted connection, use the getDB2TrustedPooledConnection object, and include the new user's password.

    ```
    getDB2TrustedPooledConnection( String user_ID,
          String user_password, java.util.Properties properties );
    ```

  - For a switch request with authentication requirements on a distributed-transaction, trusted connection, use the getDB2TrustedXAConnection object, and include the new user's password.

    ```
    getDB2TrustedXAConnection( String user_ID, String user_password,
          java.util.Properties properties );
    ```

- IBM Data Server Provider for .NET
  - For a switch request without authentication requirements, use the `TrustedContextSystemUserID` property in the connection string.

    ```
    String connectString = "
       Server = IP_address/Local_host:port_number;
       Database = database_name;
       TrustedContextSystemUserID = user_ID
    ";
    ```

  - For a switch request with authentication requirements, use the `TrustedContextSystemUserID` and `TrustedContextSystemPassword` properties in the connection string.

    ```
    String connectString = "
       Server = IP_address/Local_host:port_number;
       Database = database_name;
       TrustedContextSystemUserID = user_ID;
       TrustedContextSystemPassword = user_password;
    ";
    ```

Related reference:

SET SESSION AUTHORIZATION statement (SQL Syntax)

Rules for switching the user ID on a trusted connection

Specific rules apply to switching users on a trusted connection. Use the following rules to preserve security and auditing capability for trusted connections that are used by multiple users.

Table 4-5. Rules for switching users on a trusted connection, and potential errors related to the rules.

Switching Rule	Related Errors
The switch request must be made on a transaction boundary.	If the switch request is not made on a transaction boundary, the system rolls back the transaction, sends the switch request to the server for processing, drops the connection, and then returns an error message. SQLCODE -30020
The switch request must not come from within a stored procedure.	If the switch request is made from within a stored procedure, the system returns an error message indicating an invalid operation in the environment. The system does not drop the connection and can process subsequent requests. SQLCODE -30090
The switch request must come from a user ID that is allowed on the trusted connection.	If the switch request is made with an authorization ID that is not allowed on the trusted connection, the system drops the connection and returns an error message. SQLCODE -32509
Trusted-connection requests from user IDs that require authentication to switch must provide a correct authentication token (password).	If the trusted-context object requires authentication to switch the user ID, but the appropriate authentication token is not provided in the connection, the system drops the connection and returns an error message. SQLCODE -26456
The trusted-context object used for a trusted connection must be enabled when a switch request is made.	If the trusted-context object associated with the trusted connection is dropped or disabled and a switch request for that trusted connection is made, the system drops the connection and returns an error message. SQLCODE -26456
The new, switched user ID must hold CONNECT privileges to the database.	If the switch request is made with a user ID allowed on the trusted connection but that user ID does not hold CONNECT privilege on the database, then the system returns an error message, but does not drop the connection. SQLCODE -387

If the connection is dropped because of any of the issues previously described, the only requests acceptable by the system are:

- A COMMIT statement
- A ROLLBACK SQL statement
- A DISCONNECT request
- A CONNECT request

Pluggable authentication modules for systems running on UNIX or Linux

A Pluggable Authentication Module (PAM) is a well-defined framework for supporting different authentication modules originally developed by Sun Microsystems. PAM is supported in both 32- and 64-bit modes on Solaris, Linux, HP-UX and AIX®.

System administrators can use PAM to implement different authentication mechanisms for different applications. For example, the requirements of a system like the UNIX login program might be different from an application that accesses sensitive information from a database. PAM allows for many such scenarios in a single machine, because the authentication services are attached at the application level.

System administrators can use PAM to enable an application to select the authentication as required, and for module stacking. Many modules can be stacked one after another, thus enabling the application to be authenticated in multiple ways, before granting access. PAM provides a set of APIs to support authentication, account management, session management, and password management.

The system administrator can enable or disable the use of PAM. By default, the database server uses the traditional Informix authentication mechanism (which is based on the BSD rhosts mechanism) in order to avoid forcing major changes on users.

To use PAM with IBM Informix:

- Your Informix database server must be on an operating system platform that supports PAM.
- Your client applications must be written with a sufficiently recent version of Client SDK.
- You must have the appropriate PAM service configured in the operating system.
- You must decide which PAM authentication method provides sufficient security: the client connection password, correct input to a challenge-response prompt (for example, a RADIUS authentication server), or a combination of both.
- *Linux only:* When you configure PAM to require both password and challenge-response authentication, the PAM service always ignores the password sent in the client connection request and prompts for the password a second time.
- If you require that an application authenticate in challenge-response mode before connecting to the database server, then design the application to handle the challenge prompt.
- You must ensure that Enterprise Replication and high availability clusters are not affected by PAM authentication.
- You must modify the server entry in the `sqlhosts` file for both the client application and the database server (if they are on separate machines or in separate locations on a single machine).

Related concepts:

"Authentication mechanisms" on page 4-2

"Mapped users (UNIX, Linux)" on page 4-4

The name of the PAM service

The PAM service name identifies the PAM module.

This PAM module typically is located in the `/usr/lib/security` directory and its parameters are listed in the `/etc/pam.conf` file.

In Linux, the `/etc/pam.conf` file can be replaced with a directory called `/etc/pam.d`, where there is a file for each PAM service. If `/etc/pam.d` exists, `/etc/pam.conf` is ignored by Linux. See the system documentation for the details of this configuration file.

Authentication modes with the PAM module

The PAM module determines whether a user can authenticate by providing a password, responding correctly to a challenge, or a combination of both.

The PAM implementation in IBM Informix takes advantage of the fact that for explicit connection requests, the client sends a password to the server. You can set up PAM to make this password the only requirement for authentication to the server.

When you configure PAM to use the challenge-response protocol, authentication is complete after the user enters the correct reply to a question or other prompt. With this authentication mode, an application must be designed to respond to the challenge prompt correctly before connecting to the database server. You can set up PAM authentication to use the challenge-response mode only, so that PAM ignores the client connection password.

Linux only: If PAM is configured to authenticate users with the challenge-response protocol, the password from the client is ignored always. The PAM service on Linux prompts for the user password a second time if both password and challenge-response authentication are enabled.

PAM required stack size

You can customize the stack size available for PAM modules.

The PAM feature loads operating system or third-party PAM modules (shared libraries) into the **informix** user thread. The stack size requirements of these PAM modules cannot be predicted. For instance, on Linux some modules require more than 128 KB of stack space. Use the **PAM_STACKSIZE** configuration parameter to customize the stack size for PAM modules.

For example, set **PAM_STACKSIZE** in the `onconfig` file as follows:

```
PAM_STACKSIZE 64 # Stack size needed for the PAM modules
(K Bytes)
```

On UNIX, the default value of **PAM_STACKSIZE** is 32 KB.

On Linux, the default value is 128 KB plus the value of the **STACKSIZE** configuration parameter.

Configuring a database server to use PAM

To configure a server to use PAM, the system administrator must know:

- The name of the PAM module.
- Whether the PAM module raises a challenge in addition to accepting a simple user name and password combination.

The following `sqlhosts` file entry shows a server configured to use PAM challenge authentication:

```
ifxserver2 onsoctcp servermc portnum1 s=4,pam_serv=(pam_chal),pamauth=(challenge)
```

The following `sqlhosts` file entry shows a server configured to use PAM password authentication:

```
ifxserver2 onsoctcp servermc portnum2 s=4,pam_serv=(pam_pass),pamauth=(password)
```

LDAP authentication support on Windows

LDAP Authentication on Windows is set up and configured like the Pluggable Authentication Module (PAM) that is used on UNIX and Linux. Use the LDAP Authentication Support module when you want to use an LDAP server to authenticate your system users. The module contains source code that you can modify for your specific LDAP Authentication Support module.

The authentication module is a DLL that usually is located in the `%INFORMIXDIR%\dbssodir\lib\security` directory. The parameters of the module are listed in the `%INFORMIXDIR%\dbssodir\pam.conf` file. The source code for a fully functional LDAP Authentication Module and samples of the required configuration files are included in the `%INFORMIXDIR%\demo\authentication` directory.

The LDAP Authentication Module provides single-module authentication only. The module does not support features such as module stacking. The system administrator can enable or disable the authentication.

Related concepts:

"Authentication mechanisms" on page 4-2

Installing and customizing the LDAP authentication support module

Before you can use the IBM Informix LDAP Authentication Module to create your authentication module, you must have an LDAP server and the LDAP client-side system. Examples of LDAP systems are IBM Directory Server and openLDAP.

Your LDAP client-side system typically includes LDAP libraries and header files. These libraries and header files are required to compile the LDAP module.

To customize the LDAP Authentication Support module:

1. Customize the `pam_ldap.c` file that is included with IBM Informix.
2. Compile the `pam_ldap.c` file into a DLL and place it in a secure directory.

   Tip: Place the `pam_ldap.c` file in the `%INFORMIXDIR%\dbssodir\lib` directory.

Your installation also includes a template of a configuration file, pam_ldap_tmpl, for the LDAP module. This configuration file contains site-specific information. You must store site-specific information in this configuration file, because the file enables a single LDAP module to work in different settings.

Configuring the LDAP module

Use the template of a PAM configuration file to configure your LDAP module.

To configure your LDAP module:

1. Copy the template file to %INFORMIXDIR%\dbssodir\etc and name it pam.conf.
2. Customize the file to accommodate your local security settings. See the template file, pam.conf_tmpl, for details about how to customize the file.

Configuring IBM Informix for LDAP

To configure a server to use an LDAP Authentication Support module, edit the sqlhosts file. The system administrator must know:

- The name of the module.
- Whether the module raises a challenge in addition to accepting a simple user name and password combination.

The following sqlhosts file entry shows a server configured to use PAM challenge authentication:

```
ifxserver1 onsoctcp servermc portnum1 s=4,pam_serv=(pam_chal),pamauth=(challenge)
```

The following sqlhosts file entry shows a server configured to use PAM password authentication:

```
ifxserver2 onsoctcp servermc portnum2 s=4,pam_serv=(pam_pass),pamauth=(password)
```

Authentication mode with the LDAP module

The LDAP Authentication Support module determines whether a simple password is sufficient or other challenges are required. Implementation of the module in IBM Informix takes advantage of the fact that for explicit connections, a password is sent to the server by the client. This password can be used to satisfy the LDAP Authentication Support module in cases where a simple password is used. If the authentication mode involves responding to single or multiple challenges, the applications must be able to respond to the challenges.

Authentication module deployment

When you use authentication modules, you must consider the following issues:

- "Implicit connections with authentication modules"
- "Application development for authentication modules" on page 4-28
- "Distributed transactions and authentication modules" on page 4-30
- "Client APIs and authentication support modules" on page 4-30
- "Compatibility issues with authentication modules" on page 4-31

Implicit connections with authentication modules

Authentication responses to authentication modules, such as PAM and LDAP, expect a password. However, in implicit connections to the database server, there is no password.

PAM and LDAP are challenge oriented systems, in that the authentication response (the password) is supplied in response to a message from the authentication module. Implicit connections can work under PAM and LDAP only in challenge mode. Implicit connections in password mode result in failure.

Application development for authentication modules

The authentication method depends on the PAM or LDAP Authentication Support module installed.

The authentication method can involve challenge and response. When the PAM or LDAP Authentication Support module raises a challenge, these processes occur:

1. The database server forwards the challenge to the client.
2. The application must respond to the challenge by using a callback function that is provided by an API in the IBM Informix Client Software Development Kit (Client SDK) (Client SDK), such as the Java Database Connectivity (JDBC) Driver.
3. If the server to which the client is connecting is set up for challenge, the application must register a callback function with a Client SDK component.
4. When the Client SDK API receives a challenge from the server, the challenge is forwarded to the application by the callback function.
5. The application must respond to the challenge.
6. The Client SDK component forwards the response to the database server.

The application must be prepared to respond to multiple challenges and cannot assume the number of challenges or the challenges themselves.

The following example shows syntax of the callback function:

```
mint ifx_pam_callback(mint (*callbackfunc_ptr)(char *challenge,
 char *response, mint msg_style))
```

char **challenge*
: the character buffer in which the challenge is given by the server. The size of the buffer is fixed at 512 bytes, defined by **PAM_MAX_MSG_SIZE** in the pam_appl.h file.

char **response*
: the character buffer in which the response is provided by the user. The size of the buffer is fixed at 512 bytes, defined by **PAM_MAX_RESP_SIZE** in the pam_appl.h file.

int *msg_style*
: contains a number that indicates the type of the message given by the server. Based on the type of the response, the application can take appropriate action in the callback function.

The client application must register the callback function before making the first connection. If the callback function is not registered when the first connection is made to the database server, and the server responds, then ESQL/C returns error -1809.

The following example shows a very simple program that first registers a callback function and then unregisters it.

```
#include <stdio.h>
#include <security/pam_appl.h>

static int user_callback(char *challenge, char *response,
```

```
 int msg_style);

int main(void)
{
    EXEC SQL char passwd[]="password";
    int retval = 0;

    /* first register the callback */
    retval = ifx_pam_callback(user_callback);

    if (retval == -1)
    {
        printf("Error in registering callback\n");
        return (-1);
    }
    else
    {
        EXEC SQL database test; /* successful connection */
        /* Note that this is an implicit connection. So, the
         * application should be ready to respond to challenges.*/
        printf ("sqlcode on pam connect = %d\n", SQLCODE);
    }

    retval = ifx_pam_callback(NULL); /* unregister the callback
                                      * function */

    if (retval == -1)
    {
        printf("Error in registering callback\n");
        return (-1);
    }
    else
    {
    /* This connection throw error -1809, since the callback
    * function was unregistered statement */
         EXEC SQL database test;
        printf ("sqlcode on connect = %d\n", SQLCODE);
    }
    return 0;
}


static int user_callback(char *challenge, char *response,
 int msg_style)
{
    switch (msg_style)
    {
        /* If the msg_style is PAM_PROMPT_ECHO_OFF, the
         * application should not echo the user's response. */
        case PAM_PROMPT_ECHO_OFF:
        case PAM_PROMPT_ECHO_ON :
                printf("%s: %d:", challenge, msg_style);
                scanf("%.*s", PAM_MAX_RESP_SIZE, response);
                break;

        case PAM_ERROR_MSG:
        case PAM_TEXT_INFO:
        default:
                printf("%s: %d\n", challenge, msg_style);
               break;
    }
    return 0;
}
```

Distributed transactions and authentication modules

When IBM Informix initiates a distributed connection after the session is established, it cannot respond to authentication challenges because the timing is unpredictable. Also, the password required to connect to the local server might not be the same as the password required to connect to the remote server. Consequently, authentication for distributed connections must be completed by the remote server on the basis of trust. The remote server must trust the local server and the remote administrators must explicitly permit the user to connect from the local server to the remote server.

The sysauth table in the sysuser database on a server records the trusted remote servers and the host on which those servers run and controls incoming connections from other servers. If PAM or an LDAP Authentication Support Module is enabled in the remote servers, the system administrator can enter authorized users in the sysauth table in the sysuser database for each remote server.

Database: sysuser

Table: sysauth

Table 4-6. Schema of the sysauth table

Column	Type
username	CHAR(32)
groupname	CHAR(32)
servers	VARCHAR(128)
hosts	VARCHAR(128)

The table can contain multiple rows for a single user to permit connections from different servers and hosts. A unique index exists on the combination of username, servers, and hosts, none of which allow nulls. The groupname column must be empty; any value in the column is ignored.

For example, to permit the server to accept distributed transactions from a user known as *user1* from database server *server1* running on host *host1.example.com*:

```
insert into sysauth values ("user1", NULL, "server1", "host1.example.com");
```

For forward compatibility, ensure that each row in the table identifies one user name, one IBM Informix server name, and one host name. Do not use comma-separated or space-separated lists of server or host names in one entry.

Client APIs and authentication support modules

Only specific IBM Informix client APIs support PAM and LDAP Authentication Support modules. To use the other APIs when an authentication module is enabled on IBM Informix, you can connect to a **DBSERVERALIASES**.

The following IBM Informix client APIs support PAM and LDAP Authentication Support modules:

- ESQL/C
- ODBC
- JDBC

The other APIs do not support PAM and LDAP Authentication Support modules. To use them against a version of IBM Informix that has an enabled authentication module, connect to a **DBSERVERALIASES** that does not have the PAM parameters in the `sqlhosts` file.

The following client APIs, tools, and applications do not support PAM or LDAP Authentication Support modules:

- LibC++
- Client DataBlade API
- OLE DB
- Visual Basic Applications using ODBC
- Ilogin and ODBC Test connection
- Informix Server Administrator

For more information about ESQL/C, ODBC, and JDBC, see the *IBM Informix ESQL/C Programmer's Manual*, the *IBM Informix ODBC Driver Programmer's Manual*, and the *IBM Informix JDBC Driver Programmer's Guide*.

Compatibility issues with authentication modules

Only specific IBM Informix products support authentication modules. To use the other products when an authentication module is enabled on IBM Informix, you can connect to a **DBSERVERALIASES**.

Not all IBM Informix products and tools support PAM or LDAP authentication:

- IBM Informix-4GL does not have a mechanism for identifying callback functions and therefore cannot directly support PAM or LDAP authentication. However, if IBM Informix-4GL uses the correct version of CSDK, you can write C code that can be called from IBM Informix-4GL to handle the challenge and response protocol. To implement PAM, upgrade to the new CSDK version, modify your applications to register a callback that can handle challenges and responses, and recompile your application.
- Products such as Informix SQL do not handle the challenge and response protocol.
- The DB-Access, **dbexport**, **dbimport**, **dbload**, and **dbschema** utilities support PAM. If they receive a challenge, they pass the challenge to the user and wait for a response. This process is repeated for each challenge that the PAM module raises.
- The **onmode**, **onstat**, and **oncheck** server administration utilities do not use PAM. However, because these utilities operate on all IBM Informix ports, the utilities can function with a PAM-enabled port.
- Other server utilities do not support PAM.

If you are using any tools that do not support PAM or LDAP authentication modules, then make connections to a **DBSERVERALIASES** that does not have the PAM parameters in the `sqlhosts` file.

Simple password encryption

The simple password communication support module (SPWDCSM) provides password encryption.

This encryption protects a password when it must be sent between the client and the database server for authentication. SPWDCSM is available on all platforms.

You cannot use password encryption with encryption CSM (ENCCSM). For example, if you are using the SPWDCSM and decide to encrypt your network data, you must remove the entries for the SPWDCSM in your concsm.cfg and sqlhosts files.

You cannot use simple password CSM over a multiplexed connection.

CSM configuration file

To use a communication support module (CSM), you must have a concsm.cfg file.

An entry in the concsm.cfg file is a single line and is limited to 1024 bytes. After you describe the CSM in the concsm.cfg file, you can enable it in the **options** parameter of the sqlhosts file, as described in *IBM Informix Administrator's Guide*.

The concsm.cfg file is located in the etc directory of INFORMIXDIR by default. If you want to store the file somewhere else, you can override the default location by setting the INFORMIXCONCSMCFG environment variable to the full path name of the new location. For information about setting the environment variable INFORMIXCONCSMCFG, see the *IBM Informix Guide to SQL: Reference*.

Entries in the concsm.cfg file must conform to the following restrictions:

- The following characters are not allowed to be part of library path names:
  - = (equal sign)
  - " (double quotation mark)
  - , (comma)
- White spaces cannot be used unless the white spaces are part of a path name.

Related reference:

"Set up the concsm.cfg file for SSO" on page 4-37

Configuring password encryption

For password encryption, you must specify password encryption libraries and connection options for the simple password communication support module (CSM).

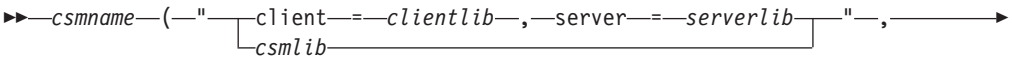
IBM Informix provides the following shared libraries for use as CSMs. The paths and fixed file names are:

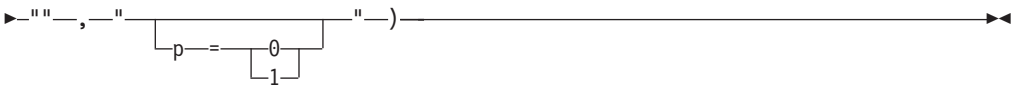
- $INFORMIXDIR/lib/client/csm/libixspw.so (UNIX and Linux)
- %INFORMIXDIR%\bin\libixspw.dll (Windows)

The shared libraries also have version-specific names that can be used in place of the fixed names. If you use the version-specific name, and the server is updated, you must update the concsm.cfg file.

To configure the CSM for password encryption, use the following syntax to add a line to $INFORMIXDIR/etc/concsm.cfg (UNIX and Linux) or %INFORMIXDIR%\etc\concsm.cfg (Windows).

Syntax





Option	Description
clientlib	The full path and name of the shared library that is the CSM on the client computer.
csmlib	The full path and name of the shared library that is the CSM if the CSM is shared by both the database server and the client computers.
csmname	The name that you assign to the CSM.
p	The password option. For CSDK version 2.3 and later, if the field is null (""), the default behavior is p=0. For CSDK before version 2.3, if the field is null, the default behavior is p=1. **1** The password is mandatory for authentication. **0** The password is not mandatory for authentication. If the client provides the password, it is encrypted and used for authentication.
serverlib	The full path and name of the shared library that is the CSM on the database server.

SMI tables and concsm.cfg setting

If you want to build the SMI tables when you open the database server (**oninit -i**), do not specify the `p=1` option in the database server CSM entry in the `concsm.cfg` file. The **oninit** process does not have a password for the **informix** or **root** user ID. If you specify the `p=1` option in the `concsm.cfg` file for the database server, you receive the following error message:

```
-5013 CSM: cannot obtain credential:
authentication error.
```

To specify that the password is mandatory for the database server CSM when the SMI tables are not yet built:

1. Do not specify the `p=1` option in the `concsm.cfg` entry.
2. Open the database server with the **oninit -i** command to build the SMI tables.
3. Bring down the database server.
4. Specify the `p=1` option in the database server CSM entry in the `concsm.cfg` file.
5. Start the database server again with the **oninit** command.

Example concsm.cfg entries for password encryption

You must specify parameters and fields in the `concsm.cfg` file for password encryption.

The following two examples illustrate the two alternatives for parameters that you must enter in the `concsm.cfg` file to define the Simple Password Communication Support Module.

The following configuration example is for a database server and client computers that use different CSMs.

```
SPWDCSM("client=/usr/informix/lib/client/csm/libixspw.so,
   server=/usr/informix/lib/csm/libixspw.so", "", "")
```

The following configuration example is for a database server and client computers that share a CSM.

```
SPWDCSM("/usr/informix/lib/csm/libixspw.so", "", "")
```

The following configuration example shows the connection option `p` set to `0`, so a password is not required:

```
SPWDCSM("/work/informix/csm/libixspw.so", "", "p=0")
```

Single sign-on authentication

Single sign-on is an authentication feature that bypasses the requirement to provide user name and password after a user logs into the client computer's operating system.

IBM Informix delivers support for single sign-on (SSO) in the Generic Security Services Communications Support Module (GSSCSM) and uses the Kerberos 5 security protocol.

With SSO, authentication for the DBMS and other SSO-enabled services happens when a user first logs into the client computer (or domain, in the case of Windows). The Kerberos implementation validates the user credentials. Kerberos authentication generates a system of secret keys that store login credentials. When a user action tries to access an Informix database, an exchange of ticket-granting tickets (TKTs) allows database access without a login prompt.

Single sign-on authentication uses both of the following open computing standards:

- Generic Security Services Application Programming Interface (GSSAPI): an API defined by Internet Engineering Task Force (IETF) standard RFC 2743 for client-server authentication
- Kerberos security protocol: RFC 1510 that defines a typical key exchange mechanism. Applications can use the Kerberos service to authenticate their users and exchange cryptographic keys containing credentials.

SSO also includes support for confidentiality and integrity services, so an SSO environment is not required to have other Informix CSMs. With confidentiality enabled in GSSCSM, the data transmitted to and from the SSO-authenticated user is encrypted and can be viewed only by the user logged in with the authorized credentials. Integrity service ensures that data sent between user and the DBMS is not altered during transmission.

GSSCSM does not function with the simple password and encryption modules (SPWDCSM and ENCCSM). SSO implemented with GSSCSM supports PAM and LDAP, but does not support mutual authentication.

Kerberos authentication protocol

For single sign-on, the user login process and authentication must employ a Kerberos 5 network infrastructure, including a dedicated Key Distribution Center computer.

A complete description of the Kerberos security protocol and how to configure it specifically for your system, are beyond the scope of this documentation. This topic orients users new to Kerberos implementations to the starting points for gathering required information.

Overview of Kerberos

Kerberos is a third-party network authentication protocol that employs a system of shared secret keys to securely authenticate a user in an unsecured network environment. The application server and client exchange encrypted keys (tickets), instead of a clear-text user ID and password pair, to establish a user's credentials on the network. A separate server called the Key Distribution Center (KDC) issues a ticket after verifying the validity of a user login.

Each user, or principal in Kerberos terms, possesses a private encryption key that is shared with the KDC. Collectively, the set of principals and computers registered with a KDC are known as a realm.

An encrypted service ticket stores a user's credentials. The database server unencrypts the ticket to verify that the credentials are associated with a user login authorized for access. While a valid service ticket exists on the network, the IBM Informix instance authorizes logged-in user access to the DBMS. The Kerberos protocol has the following security features:

- Service tickets exist on the network for a limited duration.
- Only the client and the server can unencrypt these tickets, so data is protected if the tickets are intercepted from the network.
- Input of user name and password is limited to the initial login session, reducing the risk of possible interception of clear-text credentials.

Administration of user IDs is simplified because the KDC hosts a central repository for principals. However, the disadvantage of this centralization is that it creates for a single point-of-attack by hackers. You must weigh Kerberos' advantages against this potential threat for your own environment.

Setting up an SSO authentication environment

Establishing SSO authentication for Informix involves configuration of a secured Key Distribution Center computer and connectivity files, along with generation of client and server service principals.

The overall process in deploying Kerberos SSO for Informix is as follows:

1. Configure the computers on the network to function with the Kerberos 5 authentication protocol. This involves setup of a secured computer to host the Key Distribution Center (KDC). It is possible that your network already is set up with a Kerberos mechanism.
2. Create client user principals and the Informix service principal in the KDC (see “Preparing the Informix DBMS for Kerberos authentication” on page 4-36).
3. Configure the `sqlhosts` information and Generic Security Services communications support module (GSSCSM) on the computer hosting the database server (see “Configuring an IBM Informix instance for SSO” on page 4-37).
4. Configure the Informix service principal key and ensuring it is on the computer hosting the database server.

5. Configure a database client program that functions with GSSCSM (see "Clients supporting SSO").

Related concepts:

"Mapped users (UNIX, Linux)" on page 4-4

Clients supporting SSO

Client programs that are available in the IBM Informix Client Software Development Kit (Client SDK) can connect to Informix with SSO.

See the *IBM Informix Client Products Installation Guide* for an overview of the Client SDK.

You can use the following clients with SSO:

IBM Informix ESQL/C

IBM Informix ODBC Driver

IBM Informix JDBC Driver with Sun Java Developer Kit (Sun JDK) version 1.4 onwards

IBM Informix DB-Access

See "Configuring ESQL/C and ODBC drivers for SSO" on page 4-40 and "Configuring JDBC Driver for SSO" on page 4-41 for how to set up the client programs.

Preparing the Informix DBMS for Kerberos authentication

Configure your login process and user authentication to function with a Kerberos 5 mechanism before you set up Informix for single sign-on.

Informix SSO requires installation and setup of a Kerberos 5 authentication mechanism on the client and server computers of your network. For details on setting up your network according to the Kerberos standard, see the documentation provided with the installed Kerberos product.

Important: Use a secure computer for the Key Distribution Center to ensure the safety of the passwords and encryption keys. Limit access to specific users and, if possible, do not use the computer for other tasks.

For JDBC Driver client sites, read "Configuring JDBC Driver for SSO" on page 4-41 before you do the following steps.

You must have kadmin privileges (UNIX and Linux) or domain administrator rights (Windows) to complete steps 3, 4 on page 4-37, and 5 on page 4-37.

1. For sites that are enabling a new Kerberos 5 setup for SSO, run the sample client and server programs if they are available with your Kerberos product. This task helps eliminate setup errors in the network infrastructure.
2. Verify that the clocks of all computers to be involved with SSO authentication are synchronized. Kerberos typically does not function when there is a clock discrepancy of five minutes or more between computers.
3. Create the Informix service and client principals on the Key Distribution Center (KDC) with the **kadmin** utility (UNIX and Linux) or with Active Directory (Windows). Remember the following rules as you create principals:

a. All principals to be used with Informix must be in the same realm or trusted realms.
b. All principals must map to database server user IDs. For example, if you have user5@payroll.jkenterprises as a principal, user5 must exist as an operating system user and `payroll.jkenterprises.com` as a fully qualified host name.

4. *UNIX and Linux only:* Add the server service principal key to the `keytab` file and transfer the file to the Informix host computer.
5. *UNIX and Linux only:* Put the `keytab` file into the default `keytab` file location.

Configuring an IBM Informix instance for SSO

Complete the following tasks for the server side of your system to enable SSO functionality with IBM Informix:

1. "Set SQLHOSTS information for SSO"
2. "Set up the concsm.cfg file for SSO"
3. "Ensure keytab file has the required key (UNIX and Linux)" on page 4-38
4. "Verify Informix uses Kerberos authentication for SSO" on page 4-39

Set SQLHOSTS information for SSO

This task configures the SQLHOSTS connectivity options so that your Informix instance can support single sign-on.

You must know the exact dbservername values defined in the **DBSERVERALIASES** configuration parameter before you can complete this task.

The main action of this task is to set the options field of the SQLHOSTS information to `csm=(GSSCSM)`. To modify the SQLHOSTS information:

1. Open the `sqlhosts` file (UNIX and Linux) or the SQLHOSTS registry key (Windows) on the computer hosting the database server. See the *IBM Informix Administrator's Guide* for details on how to set SQLHOSTS information.
2. Create an SQLHOSTS entry for the **DBSERVERALIASES** name that you want to use for the connection, specifying `onsoctcp` in the **NETTYPE** field and `csm=(GSSCSM)` in the OPTIONS field. For example, the following entry creates a Kerberos service for the fictional company JK Enterprises if the port number is already defined in `$INFORMIXDIR/etc/services`:

```
ol_home2data  onsoctcp  jkent-005  csm=(GSSCSM)
```

You are required to configure the SQLHOSTS information about the client computer similarly. If you are using SSO in an environment where both database server and your client program are on the same computer, then you have no other SQLHOSTS tasks to complete.

Set up the concsm.cfg file for SSO

You must specify credentials encryption libraries in the communications support module (CSM) configuration file to enable single sign-on (SSO). In addition, you can control whether SSO functions with Kerberos-defined confidentiality and integrity services.

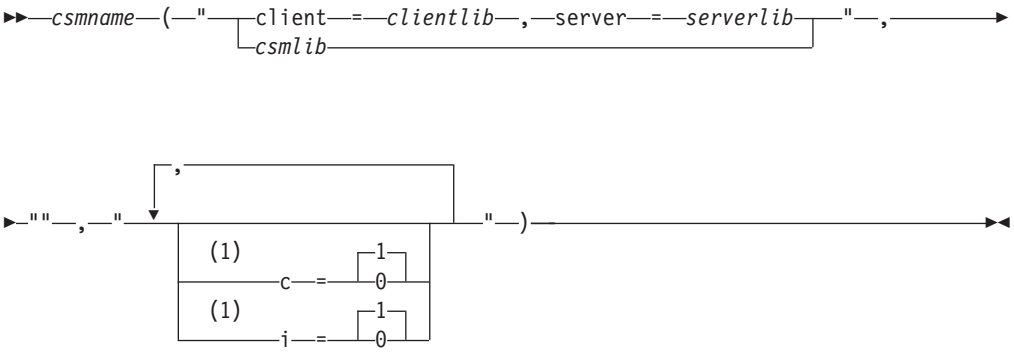
IBM Informix provides the following shared libraries for use as CSMs. The paths and fixed file names are:

- `$INFORMIXDIR/lib/csm/libixgss.so` (UNIX and Linux)
- `%INFORMIXDIR%\bin\libixgss.dll` (Windows)

The shared libraries also have version-specific names that can be used in place of the fixed names. If you use the version-specific name, and the server is updated, you must update the `concsm.cfg` file.

To configure the CSM for SSO, use the following syntax to add a line to `$INFORMIXDIR/etc/concsm.cfg` (UNIX and Linux) or `%INFORMIXDIR%\etc\concsm.cfg` (Windows).

Syntax



Notes:

1 Use path no more than once.

Option	Description
csmname	The name that you assign to the CSM
csmlib	The path and name of the shared library that is the CSM if the CSM is shared by both the database server and the client computers
clientlib	The path and name of the shared library that is the CSM on the client computer
serverlib	The path and name of the shared library that is the CSM on the database server
c	The setting for Kerberos-defined confidentiality services **0** Disabled **1** Enabled. This is the default setting
i	The setting for Kerberos-defined integrity services. **0** Disabled **1** Enabled. This is the default setting.

Related concepts:

"CSM configuration file" on page 4-32

Ensure keytab file has the required key (UNIX and Linux)

Add the service principal key generated in the Key Distribution Center to the credentials information stored in the `keytab` file on the Informix host computer, and then validate that all necessary credentials are stored in this file.

Before you can complete this task, verify that you comply with the following prerequisites:

- A valid Informix service principal has been created on the Key Distribution Center (KDC) computer. Typically, a Kerberos principal is created by using the **kadmin** utility. See your Kerberos documentation for further information.
- Client principals also exist on the KDC computer.

Important: Protect your system from intruders by maintaining appropriate security measures, such as controlling access to the keytab file.

1. Add the service principal key to the keytab file on the KDC computer.
2. Transfer the file to the keytab file for the DBMS, typically a separate computer hosting Informix.

The keytab file (UNIX and Linux):

All Kerberos server computers on UNIX and Linux must have a keytab file to authenticate with the Key Distribution Center.

A keytab file is an encrypted copy of the Informix service key. This file must be on the computer hosting Informix so that the DBMS can authenticate with the Key Distribution Center (KDC) and can accept the client's security context.

For instructions on adding a key to the keytab file, see the documentation provided with the Kerberos product.

Verify Informix uses Kerberos authentication for SSO

Before you set up the SQLHOSTS information and concsm.cfg file for the client computer in a single sign-on implementation, verify that your login service is correctly configured to use Kerberos authentication.

The client user principal and service principals must exist in the Key Distribution Center (KDC) to authenticate by using the Kerberos tickets. Also, the KDC daemon must be running.

1. Log on by using Kerberos authentication, which typically generates the required user credentials (ticket-granting ticket) for SSO on all platforms. However, if you are working on UNIX or Linux, you can also employ the **kinit** utility to obtain a ticket-granting ticket (TGT). For example, the following command can generate a TGT for the user named admin in the realm payroll.jkenterprises.com:

   ```
   % /usr/local/bin/kinit admin@payroll.jkenterprises.com
   ```

2. Use the **klist** utility to view the credentials cache from the KDC and verify the existence of a valid ticket for the user ID. A valid ticket looks similar to the following example:

   ```
   Ticket cache: FILE:/tmp/krb5cc_200
   Default principal: admin@payroll.jkenterprises.com

   Valid starting     Expires
   01/30/08 09:45:28  01/31/08 09:45:26
   Service principal
   krbtgt/payroll.jkenterprises.com@jkenterprises.com
   ```

3. After Informix accepts a connection request, verify that a valid ticket-granting service (TGS) is present. The TGS is required for the server service principal. The following example shows the output of the **klist** utility, with ol_home2data/jkent-005.payroll.jkenterprises.com as the Informix service principal.

```
Ticket cache: FILE:/tmp/krb5cc_200
Default principal: admin@payroll.jkenterprises.com

Valid starting     Expires
01/30/08 09:45:28  01/31/08 09:45:26
Service principal
krbtgt/payroll.jkenterprises.com@jkenterprises.com

01/30/08 09:48:31  01/31/08 09:45:26
ol_home2data/jkent-005.payroll.jkenterprises.com@jkenterprises.com
```

Configuring ESQL/C and ODBC drivers for SSO

The steps for preparing the SQLHOSTS information and the Generic Security Services (GSS) CSM configuration file for ESQL/C and ODBC and a client computer are similar to the corresponding server-side setup procedures.

Complete the tasks outlined in "Configuring an IBM Informix instance for SSO" on page 4-37 before working on your client.

See "Clients supporting SSO" on page 4-36 for a list of clients that support SSO with IBM Informix.

1. Complete any setup steps specific to the client software you are using. This includes the following steps:
   a. For ESQL/C, include an `sqlhosts` entry specifying `onsoctcp` in the **NETTYPE** field and `csm=(GSSCSM)` in the OPTIONS field matching the same information for the Kerberos service in the server's SQLHOSTS information. For example, the following entry can be valid for the company JK Enterprises if the port number is already set in `$INFORMIXDIR/etc/services`:

      ```
      ol_sso_krb  onsoctcp  jkent-005 ol_sso_svce csm=(GSSCSM)
      ```

   b. For ODBC on UNIX and Linux, add `Options="csm=(GSSCSM)"` to the connection settings for the SSO-enabled database server entry in the `odbc.ini` file, as illustrated in the following example:

      ```
      Driver=/usr/informix/lib/cli/iclit09b.so
      Description=IBM INFORMIX ODBC DRIVER
      Database=stores_demo
      ServerName=ol_sso_krb
      Options="csm=(GSSCSM)"
      ```

   c. *for ODBC on Windows:*
      1) Open SETNET32 (Start > All Programs > IBM Informix Client-SDK > Setnet32) and select the Server Information tab.
      2) Provide the connectivity information for the database server that is configured for Kerberos authentication. The entries in the fields of the Server Information tab correspond with the information in the SSO entry for the SQLHOSTS registry key, including `csm=(GSSCSM)` in the Options field.
      3) Open the ODBC Driver manager program.
      4) On the General tab provide your Data Source Name (DSN) details, and on the Connection tab select the SSO-enabled instance in the Server Name field.
2. Create the communications support module (CSM) configuration file for the client computer. This file must be named `$INFORMIXDIR/etc/concsm.cfg` on UNIX and Linux platforms, and `$INFORMIXDIR/etc/concsm.cfg` on Windows. Read the "CSM configuration file" on page 4-32 information for details about file requirements.

3. Add a line to `concsm.cfg` for the client computer shared libraries and for the global and connection options. See "Set up the concsm.cfg file for SSO" on page 4-37 for how to enter this configuration information.

Configuring JDBC Driver for SSO

When JDBC Driver is the client for SSO, use the DriverManager.getConnection() method, with an SSO connection property set to the Informix service principal.

Using IBM Informix JDBC Driver as the SSO client has been developed and tested with Sun Java Developer Kit (Sun JDK) 1.4 only.

1. Set the DriverManager.getConnection() method with the SSO options. The following example illustrates valid syntax for one database URL:

   ```
   ="jdbc:informix-sqli://payroll.jkenterprises.com:9555/test:
   informixserver=ol_jk_ent1;CSM=(SSO=ol_jk_ent1@jkenterprises.com,ENC=true)";
   ```

   ENC in the database URL determines whether Generic Security Services (GSS) encryption is enabled or not. By default, the setting is ENC = true (encryption enabled). See the *IBM Informix JDBC Driver Programmer's Manual* for details about the DriverManager.getConnection() method and database URLs.
2. Create a login configuration file before running the application with the following entry:

   ```
   com.sun.security.jgss.initiate {
     com.sun.security.auth.module.Krb5LoginModule required
     useTicketCache=true doNotPrompt=true;
   };
   ```

   See your Kerberos documentation about login modules for additional options.
3. Provide the login configuration file with the -D option to run the application. The following example illustrates the format for the command, where `IfmxLog.conf` is the full path and name to the login configuration file and `TestSso` is the Java class name:

   ```
   java -Djava.security.auth.login.config=IfmxLog.conf TestSso
   ```

Securing local connections to a host

The database server administrator (DBSA) can use the **SECURITY_LOCALCONNECTION** configuration parameter to set up security checking for local connections with the same host.

The following table shows the settings of the **SECURITY_LOCALCONNECTION** configuration parameter that you can use.

Table 4-7. **SECURITY_LOCALCONNECTION** *configuration parameter settings*

Setting	Explanation
0	No security checking occurs.
1	IBM Informix compares the user ID of the owner trying to connect with the connection user ID. If these do not match, IBM Informix rejects the connection.
2	IBM Informix performs the same checking that is performed when **SECURITY_LOCALCONNECTION** is set to 1. In addition, IBM Informix gets the peer port number from the network API and verifies that the connection is coming from the client program. If you set **SECURITY_LOCALCONNECTION** to 2, you must have SOCTCP or IPCSTR network protocols.

If **SECURITY_LOCALCONNECTION** is set to 1 or 2, IBM Informix establishes a connection only if the connection meets the requirements of the security check.

Limiting denial-of-service flood attacks

IBM Informix has multiple listener threads (listen_authenticate) to limit denial-of-service (DOS) attacks.

These threads authenticate client requests, while the main listener thread only accepts the incoming requests and forks new threads for authentication.

You can use the **MAX_INCOMPLETE_CONNECTIONS** configuration parameter to configure the number of the threads authenticating at any point in time.

You can use the **LISTEN_TIMEOUT** configuration parameter to configure the timeout value for incomplete connections.

DOS attacks can occur when you use external mechanisms such as Telnet to connect to the port reserved for a database server. For example, if you use Telnet to connect to the port reserved for a database server service, but do not send data, and a separate session attempts to connect to the server through an application such as DB-Access, the listener thread is blocked while waiting for information from the Telnet session and the listener thread cannot accept the connection to the application used in the second session. If during the waiting period, an attacker launches a distributed DOS (DDOS) attack in a loop, you can receive a flood attack on the connection leading to poor connection performance.

LISTEN_TIMEOUT and MAX_INCOMPLETE_CONNECTIONS configuration parameters

You can use configuration parameters to reduce the risk of a hostile, denial-of-service (DOS) flood attack.

You can customize the following configuration parameters:

- **LISTEN_TIMEOUT**. Sets the incomplete connection timeout period. The default incomplete connection timeout period is 60 seconds.
- **MAX_INCOMPLETE_CONNECTIONS**. Restricts the number of incomplete requests for connections. The default maximum number of incomplete connections is 1024.

If you do not set the **LISTEN_TIMEOUT** and **MAX_INCOMPLETE_CONNECTIONS** configuration parameters and a flood of unauthorized attacks occurs, the Listener VP might become insecure and it might not be able to listen to a valid request in a timely manner.

If you set the **LISTEN_TIMEOUT** and **MAX_INCOMPLETE_CONNECTIONS** configuration parameters, and then someone tries to break into the system and reaches the maximum limit specified, the following information in the online message log is the notification that the system is under attack:

```
%d incomplete connection at this time.
System is under attack through invalid clients
on the listener port.
```

Depending on the machine capability of holding the threads (in number), you can configure **MAX_INCOMPLETE_CONNECTIONS** to a higher value and depending on the network traffic, you can set **LISTEN_TIMEOUT** to a lower value to reduce the chance that the attack can reach the maximum limit.

You can use the **onmode -wm** or **onmode -wf** commands to change the values of these configuration parameters while the server is online. For more information, see the *IBM Informix Administrator's Reference*.

Chapter 5. Discretionary access control

Discretionary access control verifies whether the user who is attempting to perform an operation has been granted the required privileges to perform that operation.

You can perform the following types of discretionary access control:

- Create user roles to control which users can perform operations on which database objects. See "User roles."
- Control who can create databases. See "Setting permission to create databases" on page 5-2.
- Prevent unauthorized users from registering user-defined routines. See "Security for external routines (UDRs)" on page 5-3.
- Control whether other users besides the DBSA can view executing SQL statements. See "Enabling non-DBSAs to view SQL statements a session is executing" on page 5-4.

User roles

A role is a work-task classification, such as payroll or payroll manager. Each defined role has privileges on the local database object granted to the role. You use the CREATE ROLE statement to define a role.

After you create a role, use the GRANT statement to grant privileges to one or more users associated with the role name.

When a role is granted to a user, the role grantor or the role grantee (user) must use the SET ROLE statement to activate the role. Only then does the user have the privileges of the role.

Important: The scope of a role's privileges is the current database only. When the SET ROLE statement is run, the role is set in the current database only. As a security precaution, a user with role privileges cannot access tables on a remote computer through a view, trigger, or programmed procedure.

For more information about creating and using roles, see the *IBM Informix Guide to SQL: Syntax*.

Related reference:

"Example of assigning a default role in a trusted-context object" on page 4-19

"Example of assigning user-specific privileges in a trusted-context object" on page 4-20

Role separation

When you install a database server instance, you implement role separation by setting the INF_ROLE_SEP environment variable to a non-zero integer value. Role separation enforces separating administrative tasks by people who run and audit the database server. If INF_ROLE_SEP is not set, then user **informix** can perform all administrative tasks.

You cannot switch on role separation by resetting the environment after the server instance has been installed without role separation, and you cannot selectively implement role separation on only some of the databases of the same database server.

For more information about the INF_ROLE_SEP environment variable, see the *IBM Informix Guide to SQL: Reference*. For more information about role separation, see "Using role separation" on page 8-3.

Default roles

An administrator can define a default role to assign to individual users or to the PUBLIC group for a particular database.

The default role is automatically applied when a user establishes a connection with the database.

Each user has whatever privileges are granted to the user individually and the privileges of the default role. A user can switch from the current individual role to the default role by using the SET ROLE DEFAULT statement.

If different default roles are assigned to a user and to `PUBLIC`, the default role of the user takes precedence. If a default role is not assigned to a user, the user only has individually granted and public privileges.

Related reference:

"Example of assigning a default role in a trusted-context object" on page 4-19

Granting privileges for a default role

To define and grant privileges for a default role:

1. Select an existing role in the current database to use as a default role or create the role that you want to use as a default role. Use the CREATE ROLE *rolename* statement to create a new role in the current database.
2. Use the GRANT statement to grant privileges to the role.
3. Grant the role to a user and set the role as the default user or PUBLIC role by using the syntax GRANT DEFAULT ROLE *rolename* TO *username* or GRANT DEFAULT ROLE *rolename* TO PUBLIC.

Use the REVOKE DEFAULT ROLE statement to disassociate a default role from a user.

A user must use the SET ROLE DEFAULT statement to change any other current role to the default role.

See the *IBM Informix Guide to SQL: Syntax* for more information about using these statements.

Setting permission to create databases

Use the **DBCREATE_PERMISSION** configuration parameter to give specified users permission to create databases and thus prevent other users from creating databases.

If you do not set the **DBCREATE_PERMISSION** configuration parameter, any user can create a database.

The **informix** user always has permission to create databases.

To set permission to create databases:

1. To restrict the ability to create databases to the **informix** user, add the following line to the `onconfig` file:

   ```
   DBCREATE_PERMISSION informix
   ```

2. You can include multiple instances of **DBCREATE_PERMISSION** in the `onconfig` file to give additional users permission to create databases. For example, to grant permission to a user named watsonjay, add this line to the `onconfig` file:

   ```
   DBCREATE_PERMISSION watsonjay
   ```

Security for external routines (UDRs)

External routines with shared libraries that are outside the database server can be security risks. External routines include user-defined routines (UDRs) and the routines in DataBlade modules.

A database server administrator (DBSA), the user **informix** by default, can implement security measures that establish which users can register external routines. This prevents unauthorized users from registering the external routines.

Use the **IFX_EXTEND_ROLE** configuration parameter to restrict the ability of users to register external routines.

The default value of the **IFX_EXTEND_ROLE** configuration parameter is 1 (or On).

When the **IFX_EXTEND_ROLE** configuration parameter is set to On:

- You can grant a user the privileges to create or drop a UDR that has the EXTERNAL clause.
- The EXTEND role is operational and you can grant a user the privileges to create or drop an external routine that has the EXTERNAL clause.

When you grant the EXTEND role to a specific user, the **sysroleauth** system catalog table is updated to reflect the new built-in role.

After you set the **IFX_EXTEND_ROLE** configuration parameter to `On`, a DBSA can use the following syntax to grant and revoke privileges to and from specific users.

- `GRANT extend To` *username*
- `REVOKE extend From` *username*

If you do not want to restrict UDR access, set the **IFX_EXTEND_ROLE** configuration parameter to `0` (or `Off`). When the **IFX_EXTEND_ROLE** parameter is set to `Off`, the EXTEND role is not operational and any user can register external routines.

The **dbimport** utility, in particular, is affected when the **IFX_EXTEND_ROLE** configuration parameter is set to `On` because a user who uses **dbimport** to create a new database has not been given an extend role on that database.

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

Enabling non-DBSAs to view SQL statements a session is executing

The **onstat** commands that show the SQL statement text that a session is executing are normally restricted to DBSA users. You can set the **UNSECURE_ONSTAT** configuration parameter to 1 to remove this restriction. After this restriction is removed, other users can run **onstat -ses**, **onstat -stm**, **onstat -ssc**, and **onstat -sql** commands.

The **UNSECURE_ONSTAT** configuration parameter takes effect when the database server is shut down and restarted.

Chapter 6. Label-based access control

You can use label-based access control (LBAC), an implementation of multi-level security (MLS), to control who has read access and who has write access to individual rows and columns of data.

MLS systems process information with different security levels, permit simultaneous access by users with different security clearances, and allow users access only to information for which they have authorization. MLS is a well-known implementation of mandatory access control (MAC). If you hold the Database Security Administrator (DBSECADM) role in IBM Informix, you can configure the LBAC objects to meet your security requirements:

1. *Security policies.* You attach a security policy to a table that you want to protect from unauthorized access. To create a security policy, you define security labels that determine who can access the table's data. You can have one or more security policies on your system, depending on your organization's requirements.
2. *Security labels.* You associate security labels with one or more objects in a table (data labels) and with users (user labels). When a user attempts to access an LBAC-protected table object, the system compares the user label to the data label to determine if the user can have access. If the user was not granted any label, access in most circumstances is automatically blocked.
3. *Security label components.* Security label components are the building blocks of LBAC security policies. You use these components to form security policies, which, in combination with security labels, represent different user access privileges. The variety of security label components that you can create, and the flexibility that you have in constructing security policies and security labels, offers you flexibility in the way you design your organization's LBAC solution.

LBAC complements discretionary access control (DAC). When a user attempts to access a protected table, IBM Informix enforces two levels of access control. The first level is DAC. With DAC, IBM Informix verifies whether the user attempting to access the table has been granted the required privileges to perform the requested operation on that table. The second level is LBAC, which controls access at the row level, column level, or both levels. The user's credentials are a combination of DAC privileges and granted LBAC-protected data access.

This feature might not be available in all editions of Informix database server products. For details on the differences between editions, see the following website: http://www.ibm.com/developerworks/data/library/techarticle/dm-0801doe/index.html.

Configuring label-based access control

The general procedure involves a few SQL-based tasks that define precise but flexible database security objects.

Before you implement label-based access control (LBAC), you must identify the data that must to be protected, who can access that data, and what tables cannot be protected.

The following list outlines the major tasks in setting up a basic implementation with IBM Informix:

1. The database server administrator (DBSA) grants the DBSECADM role.
2. The DBSECADM defines the security objects:
   a. Creates security label components to define the attributes of sensitive data and the corresponding attributes of users who can have read access or write access to this data.
   b. Creates security policies to reflect the organization's restrictions about who can access protected data.
   c. Creates security labels for the security policies.
   d. Grants security labels to users who must have access to the protected data.
   e. *To protect new tables:* Uses the CREATE TABLE statement with the SECURITY POLICY clause and specifies how security objects protect data at the row level, column level, or at both levels.
   f. *To protect existing tables:* Uses the ALTER TABLE statement with the ADD SECURITY POLICY clause and specifies how security objects protect data at the row level, column level, or at both levels.

Tables to exclude from LBAC protection

LBAC does not protect the following categories of tables:

- virtual-table interface (VTI) tables
- tables with virtual-index interface (VII)
- temporary (TEMP) tables
- typed tables
- hierarchical tables

How security labels control access

Security labels rely on security label components to store information about the classification of data and about which users have access authority.

Label-based access control (LBAC) works by comparing the labels that you have associated with users against labels that you have associated with data by using a predefined rule set (IDSLBACRULES). You construct these labels with security label components, which represent different levels of data classification and access authority. Before you design an LBAC implementation, you must know how the labels store information in the components and how user operation and component type affect label comparison.

LBAC compares values for each user and data label when someone attempts access to a protected table. A user without a security label has a NULL value. When you create a security label, you select its values by choosing elements from each security label component that is part of the policy. Variations in the way you group the elements provide the differing values among labels that contain the same components.

LBAC compares, one-by-one, each component value of a user label to the corresponding component value in the data label. The comparison between labels is done in the sequence that the components are listed in the labels. The comparison determines if the user label component meets the appropriate IDSLBACRULE criterion for access. When all the values in the user label meet the criteria for access, the user label dominates the data label and can work with the

protected data. If any user label values do not dominate, then the user's credentials do not fit the criteria of the protecting security label. LBAC denies protected-data access to a user with a NULL value, unless the DBSECADM has granted the user an exemption to the security policy protecting the table.

When a user attempts to retrieve data from an LBAC-protected table with a SELECT operation, the comparison follows Read Access Rules.

Read Access Rules:

- IDSLBACREADARRAY: The array component of the user security label must be greater than or equal to the array component of the data security label. The user can read data only at or below the level of the value in the array component of the user label, where level is the value's relative ranking in the order of array elements.
- IDSLBACREADSET: The user security label set component must include every element in the value for the set component of the data security label.
- IDSLBACREADTREE: The tree component of the user security label must include at least one of the elements in the value for the tree component of the data security label or an ancestor of one such element.

When a user attempts an INSERT, UPDATE, or DELETE operation, the comparison follows Write Access Rules.

Write Access Rules:

- IDSLBACWRITEARRAY: The array component of the user security label must be equal to the array component of the data security label. The user can write data only at the level of the value in the array component of the user label, where level is the value's relative ranking in the order of array elements.
- IDSLBACWRITESET: The user security label set component must include every element in the value for the set component of the data security label.
- IDSLBACWRITETREE: The tree component of the user security label must include at least one of the elements in the value for the tree component of the data security label or an ancestor of one such element.

Database Security Administrator Role

The database security administrator role (DBSECADM) is required to create and maintain label-based access control security objects.

DBSECADM is a powerful server-level role that has the following responsibilities for all databases running on the IBM Informix installation:

- Create, drop, alter, and rename security label components
- Create, drop, and rename security policies
- Create, drop, and rename security labels
- Attach security policies to tables and detach security policies
- Grant security labels to users and revoke security labels
- Grant and revoke exemptions from security policies
- Grant and revoke the `SETSESSIONAUTH` privilege

Related concepts:

"Trusted-context objects and trusted connections" on page 4-14

Related tasks:

"Creating a trusted-context object" on page 4-16

Related reference:

"Requirements for trusted-context objects and trusted connections" on page 4-16

Granting the Database Security Administrator role

A DBSA uses the GRANT DBSECADM statement to give database security administrator authority to a user.

You must be a DBSA to grant DBSECADM.

Grant the DBSECADM role by issuing the GRANT DBSECADM statement, as described in the *IBM Informix Guide to SQL: Syntax*.

The following statement gives DBSECADM authority to user `sam`:

```
GRANT DBSECADM TO sam;
```

Revoking the database security administrator role

A DBSA uses the REVOKE DBSECADM statement to take away database security administrator authority from a user who previously was granted this role.

You must be a DBSA to revoke the DBSECADM role. You must know the login name from whom you want to revoke the DBSECADM role.

Use the REVOKE DBSECADM statement to revoke the DBSECADM role, as described in *IBM Informix Guide to SQL: Syntax*.

The following statement revokes DBSECADM authority from user `sam`:

```
REVOKE DBSECADM FROM sam;
```

Security label components

Security label components are security objects for defining security policies. The elements of these components are used to define security labels, which control access to protected tables.

Security label components represent any criteria that your organization might use to decide if a user must have access to a table row or column. Typical examples of such criteria include:

- How much authority the user has in the organization
- Which confidential data, if any, the user is entitled to read or write
- To which department the user belongs
- Whether the user is involved in a particular project

Before you create security label components, you must know how your organization's privacy plan corresponds with a data classification scheme. You also must identify the security policy and security labels that you build from the components. Data classifications that you implement through label-based access control (LBAC) map to the elements that you list when you create security label

components. When a user attempts to access protected data, the label values of a user is compared to the label values of the row or column. Security label components, and their elements that are used in the security labels, specify these values.

Types of security label components

There are three types of security label components:

- ARRAY: Each element represents a point on an ordered scale of relative values (see “Security label component type: ARRAY” on page 6-6)
- SET: Each element represents one member of an unordered set (see “Security label component type: SET” on page 6-7)
- TREE: Each element represents a node in a tree-like hierarchy (see “Security label component type: TREE” on page 6-7)

As you design an LBAC solution, you identify the security label component type that best reflects the relationship among varying authority levels and groups of users. A basic LBAC implementation can draw on the organization's existing categorizations to name and group the elements, so that the elements are entities the organization already uses. As an overview, the following examples briefly describe the way security label components can function in two different situations.

Example of a component reflecting a strictly ranked data classification scheme

If you are creating a security label component to represent a simple, linear ranking of data-access classifications, you use a component of type ARRAY. An ARRAY-type security label component that represents four data-access classifications can have the following elements: Top Secret, Secret, Confidential, and Unclassified.

Example of a component reflecting an organizational chart

The executive management of a fictional information-services corporation in the United States named "JK Enterprises" wants to limit access to specific rows of data on a database to which all employees have access. JK Enterprises has branched its national organization into regions and subregions. Much of JK Enterprises' privacy policy to be implemented with LBAC allows or denies access based on the user's affiliation with a regional level. The higher-level regions encompass larger areas of the organization.For example, an employee designated as part of the West regional level is entrusted with more authority than employees designated with the subordinate Southwest, California, and Pacific Northwest regional levels. The security label component type that best suits this set of criteria is TREE. Therefore, the user with DBSECADM authority at JK Enterprises creates a security label component named `region` and identifies the following elements for the component:

- `West`
- `Southwest`
- `California`
- `Pacific Northwest`

Because the regions of JK Enterprises encompass the entire United States, the four regions previously listed compose a partial list of elements. The diagram in “Security label component type: TREE” on page 6-7 illustrates all the elements of

this company's`region` security label component.

Security label component type: ARRAY

Security label component type ARRAY represents a ranked group of elements.

The elements in an ARRAY component represent an ordered scale of relative values; the first element listed has the highest value and the last has the lowest.

The maximum number of elements in an ARRAY type of security label component is 64. An ARRAY component of a security label has the value of only one of its elements when it is compared after the IDSLBACRULES.

Example: If the fictional company JK Enterprises defines security label component `level` as a ranking of the company's four different privacy levels, as in the following statement:

```
CREATE SECURITY LABEL COMPONENT level
   ARRAY [ 'Top Secret', 'Secret', 'Confidential', 'Unclassified' ];
```

Then Figure 6-1 illustrates the order of the elements:

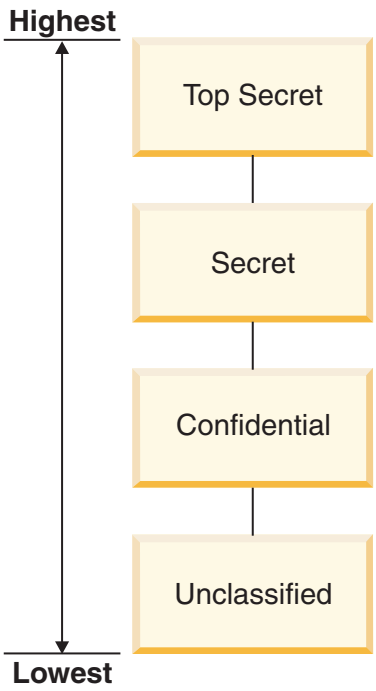


Figure 6-1. Relationship of elements in an ARRAY example

When an ARRAY component in the user label is compared to an ARRAY component of a data label:

- *For Read Access*: The IDSLBACREADARRAY rule lets the user component dominate when its value is greater than or equal to the value of the component in the data security label. The user can read data only at or below the level of the value in the array component of the user label.
- *For Write Access:* The IDSLBACWRITEARRAY rule lets the user component dominate when its value is equal to the value of the component in the data label. The user can write data only at the level of the value in the array component of the user label.

Security label component type: SET

Security label component type SET is used to represent a group of unordered elements.

A SET-type security label component consists of an unordered list of elements. There is no ranking or other relationship among elements in this type of component.

The maximum number of elements that can exist in a SET is 64. The value of a SET-type component in a security label can consist of one or more elements.

The following statement creates a SET-type security label component named `function` with three elements:

```
CREATE SECURITY LABEL COMPONENT function
SET {'Developer', 'Administrative', 'Legal'};
```

When a SET component in the user label is compared to a SET component of a data label:

- For read access, the IDSLBACREADSET rule lets the user label dominate when the SET component of the user security label includes all the elements of the value for the SET component of the data security label.
- For write access, the IDSLBACWRITESET rule lets the user label dominate when the SET component of the user security label includes all the elements of the value for the SET component of the data security label.

Security label component type: TREE

Security label component type TREE contains a group of elements that represent a family of parent-child relationships.

The elements in this type of security label component can be thought of as being in a tree. The first element you specify for a TREE-type component is ROOT, which represents the highest level of authority. Then you specify the other elements sequentially to follow the different levels of children and grandchildren that you want in the component.

The maximum number of elements in a TREE security label component is 64. The value of a TREE component in a label can be one or more of its nodes.

Example: JK Enterprises decides that its levels of authority to access protected data must follow its organizational chart. The company can use this scheme to outline its TREE security label component. The following example shows a statement creating the `region` security label component:

```
CREATE SECURITY LABEL COMPONENT region
TREE ( 'USA Headquarters' ROOT,
       'West' UNDER 'USA Headquarters',
       'Central' UNDER 'USA Headquarters',
       'East' UNDER 'USA Headquarters',
       'Pacific Northwest' UNDER 'West',
       'California' UNDER 'West',
       'Pacific Southwest' UNDER 'West',
       'North Central' UNDER 'Central',
       'South Central' UNDER 'Central',
       'Northeast' UNDER 'East',
       'Mid Atlantic' UNDER 'East',
       'Southeast' UNDER 'East');
```

Figure 6-2 illustrates the relationships among the TREE component elements in this example.

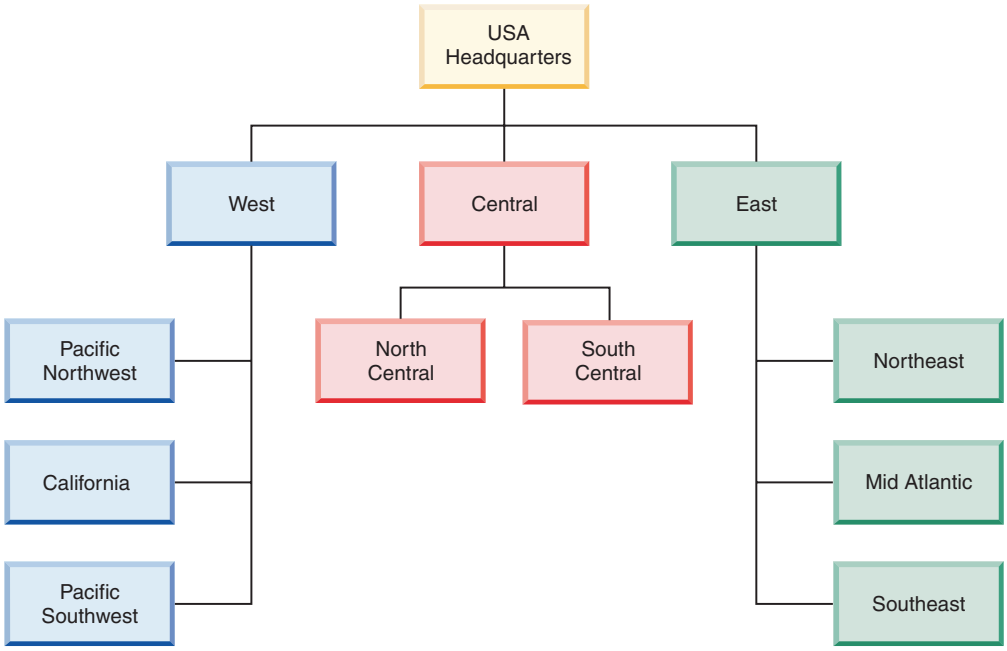


Figure 6-2. Relationship of elements in a TREE example

When a user label with one or more TREE components is compared to a data label with TREE components:

- e
- For read access, the IDSLBACREADTREE rule lets the user label dominate and have read access when the label's TREE component includes at least one of the elements in the value for the tree component of the data label or the ancestor of one such element.
- For write access, the IDSLBACWRITETREE rule lets the user label dominate and have write access when each of the label's TREE components includes at least one of the elements in the value for the tree component of the data label or the ancestor of one such element.

Creating security label components

The CREATE SECURITY LABEL COMPONENT statement defines this database security object.

You must hold the DBSECADM role to create security label components.

When you create a security label component you must provide the following information:

- A name for the component
- The type of component it is (ARRAY, SET, or TREE)
- A complete list of elements

Create a security label component by issuing the CREATE SECURITY LABEL COMPONENT statement, as described in *IBM Informix Guide to SQL: Syntax*.

The following example shows a CREATE SECURITY LABEL COMPONENT statement that creates a SET-type component with name `department` and elements `Marketing`, `HR`, and `Finance`:

```
CREATE SECURITY LABEL COMPONENT department
   SET {'Marketing', 'HR', 'Finance'};
```

Altering security label components

The ALTER SECURITY LABEL COMPONENT statement adds one or more new elements to an existing component.

You must hold the DBSECADM role to add one or more elements to a security label component, and you must know what type of component it is.

When you alter a security label component, remember the following rules:

- A security label component can consist of no more than 64 elements.
- If the component you want to alter is of type ARRAY or TREE, you must know the relationships that all the elements of the resulting component have with one another.
- The ALTER SECURITY LABEL COMPONENT statement cannot modify or drop any existing elements.

Add one or more elements to a security label component by issuing the ALTER SECURITY LABEL COMPONENT statement, as described in *IBM Informix Guide to SQL: Syntax*.

The following example shows an ALTER SECURITY LABEL COMPONENT statement that adds to a SET-type component the elements `Training`, `QA`, and `Security`:

```
ALTER SECURITY LABEL COMPONENT department
   ADD SET {'Training', 'QA', 'Security'};
```

Security policies

Security policies are database objects that you create and use to protect tables from unauthorized access.

A security policy is a named database object defined by a group of security label components.

A security policy is attached to one or more tables to allow only users with valid label-based access control credentials to read or write protected data. A user has valid credentials when the user has a security label that dominates when compared to a labeled row or column after the IDSLBACRULES. A security policy has no effect on data that has no security label.

No more than one security policy can be attached to a table, and a security policy can include no more than 16 security label components. You attach a security policy to a table by using a clause in a CREATE TABLE or ALTER TABLE statement. See “Protecting tables at row and column levels” on page 6-12 for how to attach the policy to a table.

Creating security policies

You create security policies after you have created security label components.

You must hold the DBSECADM role to create a security policy. The maximum number of security label components with which you can build a security policy is 16.

The order in which you list security label components when you create a security policy does not indicate any sort of precedence or other relationship among the components, but it is important to know the order when creating security labels with built-in SECLABEL functions.

Create a security policy by issuing the CREATE SECURITY POLICY statement, as described in *IBM Informix Guide to SQL: Syntax*.

The following example shows this SQL statement, where `company` is the security policy name and `region` and `department` are security label components used in the policy:

```
CREATE SECURITY POLICY company
   COMPONENTS region, department;
```

Security labels

Security labels are objects applied to rows and columns in order to protect these data, and granted to users to give them access to protected data.

When users try to access protected data, label-based access control compares the user label to the data label. The process of this comparison is detailed in "How security labels control access" on page 6-2.

When you create a security label:

- You identify to what security policy the label belongs.
- You assign a value for each security label component in the security policy.

You apply just one label to a row or column. For a given security policy, you typically grant to a user one label to define both read and write access. But you can grant a user one label for read access and a different label for write access to data protected by the same security policy. When the read-access label differs from the write-access label granted to a user, the user can only write to data objects that can be accessed by the user's read-access label.

Creating security labels

The CREATE SECURITY LABEL statement defines a new security label for a specified security policy.

You must hold the DBSECADM role to create a security label.

When you create a security label, you complete the following steps:

- Specify a security policy to which the label belongs.
- Identify the components of that policy.
- Identify one or more elements of each component.
- Name the label.

Create a security label by issuing the CREATE SECURITY LABEL statement, as described in *IBM Informix Guide to SQL: Syntax*.

The following example shows a CREATE SECURITY LABEL statement:

```
CREATE SECURITY LABEL company.label2
   COMPONENT level 'Secret',
   COMPONENT function 'Administrative',
   COMPONENT region 'Southwest';
```

This statement defines `label2` in security policy `company`.

Granting security labels

The GRANT SECURITY LABEL statement grants a security label to a user or to a list of users.

You must hold the DBSECADM role to grant a label to users. Users specified in a GRANT SECURITY LABEL statement cannot be the DBSECADM who issues it.

When you issue the GRANT SECURITY LABEL statement, you can optionally specify that the users receive the label for read access, write access, or all access. If you do not specify access, then the statement grants users an all-access label.

If a user is granted a different security label for read access than for write access, then the values given for the security label components must follow these rules:

- For security label components of type ARRAY, the value must be the same in both security labels.
- For security label components of type SET, the values given in the security label used for WRITE access must be a subset of the values given in the security label used for READ access. If all of the values are the same, this is considered a subset, and is allowed.
- For security label components of type TREE, every element in the TREE component of the security label for write access must be either an element or a descendent of an element in the TREE component of the security label for read access.

To grant a security label, see the documentation about the GRANT SECURITY LABEL statement in *IBM Informix Guide to SQL: Syntax*

In the following example of this SQL statement, `label2` of the `company` security policy is granted to user `maria`.

```
GRANT SECURITY LABEL company.label2
   TO maria;
```

Revoking security labels

The REVOKE SECURITY LABEL statement revokes a security label from a user or from a list of users.

You must hold the DBSECADM role to issue the REVOKE SECURITY LABEL statement.

When you issue the statement, you optionally can also:

- Revoke every security label of a security policy from users.
- Specify read-access or write-access label, or both labels, if the users have two different labels for a security policy.

Revoke a security label by issuing the REVOKE SECURITY LABEL statement, as described in *IBM Informix Guide to SQL: Syntax*.

In the following example of this SQL statement, label2 of the company security policy is revoked from user maria.

```
REVOKE SECURITY LABEL company.label2
   FROM maria;
```

Security label support functions

Security label support functions are expressions for manipulating security labels.

You typically use the security label support functions (SECLABEL functions) to specify a label in data-manipulation (DML) operations on protected table rows. In these operations, however, the security label support functions do not provide any more access to protected data than is already provided by your security credentials. There are three built-in functions for label-based access control in IBM Informix:

- The SECLABEL_BY_NAME function provides a security label directly by specifying its name.
- The SECLABEL_BY_COMP function provides a security label directly by specifying its component values.
- The SECLABEL_TO_CHAR function returns a security label in the security label string format.

You can reference a security label with these functions by providing one of the following pieces of information:

- A name, as declared in the CREATE SECURITY LABEL or RENAME SECURITY LABEL statement.
- A list of values for each component of the security policy of the security label.
- An internal encoded value that the IDSSECURITYLABEL data type stores.

These functions can convert between the various forms of a security label.

See the *IBM Informix Guide to SQL: Syntax* for more information about and examples of security label support functions.

Protecting tables at row and column levels

Protect rows and columns by associating them with security objects by including clauses in the CREATE TABLE and ALTER TABLE statements.

After you have created the security objects required for your label-based access control (LBAC) implementation, you must apply them to the tables that you want to protect. The main actions to protect the data at this stage are:

- Attach a security policy to each table containing data to be protected by LBAC.
- Associate the necessary rows and columns with security labels.

Data in a table can only be protected by security labels that are part of the security policy protecting the table. Data protection, including attaching a security policy to a table, can be done when creating the table or later by altering the table.

Protected table with row-level granularity

A table can be marked as protected with row-level granularity during CREATE TABLE or ALTER TABLE by attaching a security policy and by specifying the security label column. The security label column must be of the IDSSECURITYLABEL data type.

If users attempt to access a row to which they do not have the required LBAC credentials, the system responds to the users as if the row did not exist.

Protected table with column-level granularity

A database table can be marked as protected with column-level granularity during CREATE TABLE or ALTER TABLE by attaching a security policy to such table and by attaching a security label to one or more columns of that table. When a column is associated with a security label, that column becomes a protected column. The security policy attached to the table affects what security label can be applied to the column.

If users attempt to access a column to which they do not have the required LBAC credentials, the system generates an error message.

Security label column (IDSSECURITYLABEL data type)

The column holding the label for row-level granularity must be of the IDSSECURITYLABEL data type. Only a user who holds the DBSECADM role can create, alter, or drop a column of this data type. ISDSECURITYLABEL is a built-in DISTINCT OF VARCHAR(128) data type. A table that has a security policy can have only one IDSSECURITYLABEL column.

The following constraints cannot be applied to a security label column:

- Referential constraints
- Check constraints
- Primary key or unique constraints if the security label column is the only column in constraint
- Column protection
- Encryption

For more information about the IDSSECURITYLABEL data type, see the *IBM Informix Guide to SQL: Reference* and *IBM Informix Guide to SQL: Syntax*.

Simultaneous row-level and column-level protection on a table

A protected table can be defined with both row and column-level granularities. If both row and column granularity are applied to a table, then LBAC enforces column-level before row-level access control.

You can apply row and column-level protection on a table in a single statement rather than issuing separate statements for the two granularities when you do the either of the following steps:

- When you create a new LBAC-protected table
- When you alter a table to add row-level protection in addition to the existing column-level protection

The following example shows a CREATE TABLE statement and an ALTER TABLE statement that set up two tables with both row and column-level protection.

```
CREATE TABLE T5
   (C1 IDSSECURITYLABEL,
   C2 int,
   C3 char (10) COLUMN SECURED WITH label6)
   SECURITY POLICY company;

ALTER TABLE T6
   ADD (C1 IDSSECURITYLABEL),
   MODIFY (C2 INT COLUMN SECURED WITH label7),
   ADD SECURITY POLICY company;
```

For more information about how these statements work, see "Applying row-level protection," "Applying column-level protection," *IBM Informix Guide to SQL: Reference*, and *IBM Informix Guide to SQL: Syntax*.

Applying row-level protection

Protect row-level data by associating the table with a security policy and inserting an IDSSECURITYLABEL-type column.

There are two methods for applying row-level protection:

1. For a new table: Use the CREATE TABLE statement with the appropriate IDSSECURITYLABEL and SECURITY POLICY clauses, as described in *IBM Informix Guide to SQL: Syntax*.
2. For an existing table: Use the ALTER TABLE statement with the appropriate IDSSECURITYLABEL and ADD SECURITY POLICY clauses, as described in *IBM Informix Guide to SQL: Syntax*.

The following example shows a statement that applies row-level protection when you create a new table (`T1`) by using the security policy named `company` and the security label named `label2`.

```
CREATE TABLE T1
   (C1 IDSSECURITYLABEL,
   C2 int,
   C3 char (10))
   SECURITY POLICY company;
```

The following statement provides an example of applying row-level protection on a table (`T2`) that already exists on the database, by using the security policy named `company`. The default value for C1 is `label3`.

```
ALTER TABLE T2
  ADD (C1 IDSSECURITYLABEL DEFAULT 'label3'),
  ADD SECURITY POLICY company;
```

Applying column-level protection

Protect column-level data by associating the table with a security policy and attaching a security label to one or more columns.

There are two methods for applying column-level protection:

1. For a new table: Use the CREATE TABLE statement with the COLUMN SECURED WITH and SECURITY POLICY clauses, as described in *IBM Informix Guide to SQL: Syntax*.
2. For an existing table: Use the ALTER TABLE statement with the MODIFY (*your_column* COLUMN SECURED WITH) and ADD SECURITY POLICY clauses, as described in *IBM Informix Guide to SQL: Syntax*.

The following example shows a statement that applies column-level protection when a new table (T3) is created by using the security policy named company and a security label named label4.

```
CREATE TABLE T3
   (C1 CHAR (8),
   C2 int COLUMN SECURED WITH label4,
   C3 char (10))
   SECURITY POLICY company;
```

The following statement provides an example of applying column-level protection on a table (T4) that already exists on the database by using the security policy named company and a security label named label5.

```
ALTER TABLE T4
   MODIFY (C1 CHAR (8) COLUMN SECURED WITH label5),
   ADD SECURITY POLICY company;
```

Exemptions

Exemptions modify security credentials of users by disabling one or more of the IDSLBACRULES for a component type in a security policy.

Since exemptions are based on a security label component type for a particular security policy, this exemption does not apply outside that security policy. Within the security policy, the exemption applies to all instances of the component type.

Exemptions can be useful for allowing trusted users do administrative work for which otherwise it would be cumbersome to grant all necessary label-based access control credentials. For example, if your job is to classify incoming data, a typical practice would be for the DBSECADM to grant you exemptions so that you can write to any data row in the security policy.

If users hold an exemption to every rule of a security policy, then they have complete access to all data protected by that policy.

Exemptions provide very powerful access. Do not grant them without careful consideration.

Granting exemptions

The GRANT EXEMPTION statement gives a user an exemption from one or more access rules of a security policy.

You must hold the DBSECADM role to grant exemptions.

Grant an exemption by issuing the GRANT EXEMPTION statement, as described in the *IBM Informix Guide to SQL: Syntax*.

The following statement grants user maria an exemption from the IDSLBACWRITETREE rule in security policy company:

```
GRANT EXEMPTION
ON RULE IDSLBACWRITETREE
FOR company
TO maria
```

To grant a user exemptions from all IDSLBACRULES of a security policy, specify `ALL` in place of the policy name in the statement. Typically, this type of exemption is practical for a user who is responsible for loading and unloading data in protected tables.

Revoking exemptions

The REVOKE EXEMPTION statement revokes from a user an exemption on one or more access rules of a security policy.

You must hold the DBSECADM role to revoke exemptions.

Revoke an exemption by issuing the REVOKE EXEMPTION statement, as described in the *IBM Informix Guide to SQL: Syntax*.

The following statement revokes from user `maria` an exemption from the IDSLBACWRITETREE rule in security policy `company`:

```
REVOKE EXEMPTION ON RULE IDSLBACWRITETREE FOR company FROM maria
```

To revoke all IDSLBACRULES exemptions that a user has for a security policy, specify `ALL` in place of the policy name in the statement.

Maintaining a label-based access-control implementation

Optimizing database performance can require adjusting the values of configuration parameters for security policies and user credentials.

LBAC cache tuning

Poor performance of a database with tables protected by label-based access control (LBAC) can indicate that the system is unnecessarily relying on disk operation more than on LBAC-related caching to retrieve information from memory.

Fine-tuning one or more of the following parameters in the `onconfig` file can improve performance for queries frequently run on protected tables. For example, if the value for the **PLCY_HASHSIZE** parameter is set too low, there are not enough hash buckets allocated for security policy information caching and consequently some database performance involving LBAC-protected tables declines.

PLCY_HASHSIZE

The **PLCY_HASHSIZE** parameter specifies the number of hash buckets in the security policy information cache.

onconfig.std value
: 31

units
: KB

range of values
: Any positive integer

takes effect
: When the database server is shut down and restarted

see
: *IBM Informix Performance Guide* for information about configuration effects on memory

PLCY_POOLSIZE

The **PLCY_POOLSIZE** parameter specifies the maximum number of entries in each hash bucket of the security policy information cache.

default onconfig.std value
127

units
KB

range of values
16 or higher positive integer value

takes effect
When the database server is shut down and restarted

see
IBM Informix Performance Guide for information about configuration effects on memory

USRC_HASHSIZE

The **USRC_HASHSIZE** parameter specifies the number of hash buckets in the LBAC credential memory cache. This memory cache holds information about users' LBAC credentials.

onconfig.std value
31

units
KB

range of values
Any positive integer

takes effect
When the database server is shut down and restarted

see
IBM Informix Performance Guide for information about configuration effects on memory

USRC_POOLSIZE

The **USRC_POOLSIZE** parameter specifies the maximum number of entries in each hash bucket of the LBAC credential memory cache. This memory cache holds information about users' LBAC credentials.

default onconfig.std value
127

units
KB

range of values
16 or higher positive integer value

takes effect
When the database server is shut down and restarted

see

IBM Informix Performance Guide for information about configuration effects on memory

Dropping security objects

Use the DROP SECURITY statement to remove a security label component, a security policy, or a security label from the database.

You must hold the DBSECADM role to remove a security object.

Three valid keyword definitions of the DROP SECURITY statement are as follows:

1. DROP SECURITY POLICY *policy* removes a security policy; this can be used in RESTRICT and CASCADE modes
   - Example: `DROP SECURITY POLICY company` removes the policy named `company` from the database
2. DROP SECURITY LABEL *policy.label* removes a security label; this can be used in RESTRICT mode
   - Example: `DROP SECURITY LABEL company.label2` removes the label named `label2`.
3. DROP SECURITY LABEL COMPONENT *component* removes a security label component; this can be use in RESTRICT mode
   - Example: `DROP SECURITY LABEL COMPONENT department` removes the component `department`.

   For more information about the DROP SECURITY statement, including details about the RESTRICT and CASCADE modes, see *IBM Informix Guide to SQL: Syntax*.

When the DROP SECURITY statement executes successfully, the database server deletes any rows that reference the name or the numeric identifier of the specified object from the tables of the system catalog, including the following tables:

- **sysecpolicies** for security policies
- **sysseclabels** for security labels
- **sysseclabelcomponents** for security label components

Renaming security objects

Use the RENAME SECURITY statement to rename a security policy, a security label, or a security label component.

You must hold the DBSECADM role to rename a security object.

The three valid clauses for the RENAME SECURITY statement are as follows:

1. POLICY *old_name* TO *new_name* renames a security policy
   - Example: `RENAME SECURITY POLICY company TO subsidiary;` renames the policy named `company` to `subsidiary`
2. LABEL *security_policy.old_name* TO *new_name* renames a security label; in this statement you also indicate the security policy to which the label belongs
   - Example: `RENAME SECURITY LABEL subsidiary.label8 TO label9;` renames `label8` to `label9`, which belongs to security policy `subsidiary`
3. LABEL COMPONENT *old_name* TO *new_name* specifies a security label component

- Example: `RENAME SECURITY LABEL COMPONENT department TO division;` renames the component `department` to `division`.

For more information about the RENAME SECURITY statement, see *IBM Informix Guide to SQL: Syntax*.

The RENAME SECURITY statement replaces the *old_name* with the specified *new_name* in the table of the system catalog in which the renamed security object is registered:

- sysecpolicies.*secpolicyname* for security policies
- sysseclabels.*seclabelname* for security labels
- sysseclabelcomponents.*compname* for security label components.

This statement does not, however, change the numeric value of the sysecpolicies.secpolicyid, sysseclabels.seclabelid, or sysseclabelcomponents.compid of the renamed security object.

IBM Informix security considerations for label-based access control

The wide range of IBM Informix capabilities requires certain precautions and planning to ensure protected tables can be accessed appropriately.

The following actions require holding the DBSECADM role after you have implemented label-based access control (LBAC) on your database server:

- Using the SETSESSIONAUTH privilege (see "SET SESSION AUTHORIZATION statement")
- Exporting schema and data (see "The dbschema, dbexport, and dbimport Utilities" on page 6-20)
- Importing data (see "The dbschema, dbexport, and dbimport Utilities" on page 6-20)

These actions require the user to have read and write access credentials:

- Backing up and restoring with **onbar** and **ontape** utilities (see "Backup and restore" on page 6-20)
- "Data loading and unloading" on page 6-20

To prevent unauthorized access to protected tables, take extra precautions with the following database operations and objects:

- "The onlog utility" on page 6-21
- "The oncheck utility" on page 6-21
- "Enterprise replication" on page 6-21
- "Data definition language (DDL) operations" on page 6-21
- "INSERT INTO . . . SELECT FROM Statement" on page 6-21
- High Performance Loader .RET and .FLT files (see "Data loading and unloading" on page 6-20)
- "Temporary tables created by the INTO TEMP clause" on page 6-22
- "User-defined routines" on page 6-22 created with DBA keywords

SET SESSION AUTHORIZATION statement

You can use The SET SESSION AUTHORIZATION statement to assume the identity of another user, including the user's LBAC credentials for protected tables.

IBM Informix 11.10 and later versions that have label-based access control (LBAC) capability handles the SETSESSIONAUTH privilege differently from earlier versions of the database server that did not have LBAC functionality. The newer versions of IBM Informix require the DBSECADM to grant the SETSESSIONAUTH privilege. Because the SETSESSIONAUTH privilege can be used to assume the LBAC credentials of another user, the DBSECADM must be careful in granting the SETSESSIONAUTH privilege.

If the database server has been converted from an earlier version that did not support LBAC, users who held the DBA privilege are automatically granted the SETSESSIONAUTH access privilege for PUBLIC in the migration process. You must initialize the converted server as a version that supports LBAC security policies to remove the SETSESSIONAUTH privilege from all DBAs and enable the DBSECADM role to grant this privilege.

For more information about how SET SESSION AUTHORIZATION operates with LBAC, see *IBM Informix Guide to SQL: Syntax*.

Backup and restore

Users who are responsible for backing up or restoring protected data with an **onbar** and **ontape** utilities must have LBAC read-and-write access credentials for the corresponding server tables. LBAC security remains intact during backup and after being restored on the server, but to protect the saved backup data you must take other precautions.

You can restore of a specific table or set of tables that have previously been backed up with **onbar** or **ontape**. These tables can be restored to a specific point in time. During table-level restore of LBAC-protected tables, ensure that the schema command files specify the security policy with the target table. Because a protected target table is created during the restore, the user running the table level restore must hold the DBSECADM role. Also, LBAC rules are enforced when the INSERT statement from the schema command file is executed to load the target table. If the entire table is to be restored, the user must possess the necessary LBAC credentials.

You cannot use the **archecker** utility to perform a table-level restore.

The dbschema, dbexport, and dbimport Utilities

LBAC rules are enforced on protected tables when the **dbschema** and **dbexport** utilities are run. Only those rows are unloaded where the user's security label dominates the column label, row label, or both. Since both **dbschema** and **dbexport** utilities must read LBAC catalogs, the user running these utilities must have the appropriate LBAC credentials or exemptions to access the data.

The **dbimport** utility creates and populates a database from text files. The user importing LBAC-protected data with this utility must have the DBSECADM role. After the import process is complete, the DBSECADM role does not have any exemptions that were defined before the import process.

Data loading and unloading

IBM Informix provides a number of ways to load and unload data. Some of these methods are:

- **dbload** utility

- **onpload** and **ipload** utilities for High-Performance Loader (HPL)

LBAC rules are applied when these statements are executed, or utilities are run, on protected tables. The user's security label must dominate the column label, row label, or both. If an entire table is to be loaded/unloaded, then the user must have the necessary LBAC credentials to read and write all the labeled rows and columns. Alternatively, the DBSECADM can grant an exemption to the user so that the security policy protecting the tables can be bypassed.

Rows that are rejected when the **onpload** utility is run are dumped to .REJ and .FLT files. Take the necessary precautions to prevent unauthorized access to these files. For express-mode loads using HPL, the rows are inserted directly to table extents skipping the SQL layer. The user running the express-mode load must be granted the necessary exemptions to bypass the security policy.

You cannot use the **onload** and **onunload** utilities with LBAC.

The onlog utility

The **onlog** utility displays all or selected portions of the logical log. This command can take input from selected log files, the entire logical log, or a backup tape of previous log files. The log records can expose data that is protected by LBAC on a live database. Take precautions to ensure data is not exposed by misuse of this utility.

The oncheck utility

The **oncheck** utility can display pages from tables or chunks, which can expose data that is protected by LBAC on a live database. Take precautions to ensure data is not exposed by misuse of this utility.

Enterprise replication

You cannot apply LBAC to a table participating in Enterprise Replication. Also, you cannot define an Enterprise Replication replicate on a table that is protected by LBAC.

Data definition language (DDL) operations

LBAC does not restrict users on your system from performing data definition language (sometimes called *data definition statements*) operations. For example, a user whom has not been granted security policy credentials or an exemption can run TRUNCATE TABLE or DROP TABLE on an LBAC-protected table.

INSERT INTO . . . SELECT FROM Statement

When the INSERT INTO . . . SELECT FROM statement is used on an LBAC-protected table to create another table, ensure that the new table is protected by the same security policy used to protect the source table. Otherwise, the new table can potentially expose data in violation of your organization's privacy policy. Note that this potential data exposure can happen if the statement is used to create a permanent table, or to create a temporary table and then inserted into a permanent one.

Temporary tables created by the INTO TEMP clause

The INTO TEMP clause of the SELECT statement creates a temporary table to hold the query results. If the table being selected from is a protected table, the query-result data in the intermediate temporary table is not protected by LBAC. Take the necessary precautions to ensure that the data in the temporary table is not exposed to unauthorized users.

User-defined routines

You can register user-defined routines (UDRs) with the DBA keyword. If a user is granted the execute privilege on a UDR, the database server automatically grants the user temporary DBA privileges that are enabled only when the user is executing the UDR. The user executing the DBA UDR assumes the identity of a DBA for the duration of the UDR has the DBA's user label during that time. Avoid using protected tables in DBA UDRs.

Other IBM Informix functionality with label-based access control

IBM Informix has non-security functionality that operates seamlessly with label-based access control.

IBM Informix label-based access control (LBAC) is designed to work smoothly with all parts of the database server and without excessive user intervention to contain unauthorized data exposure. The following areas of IBM Informix are highlighted to address potential areas of concern.

High-availability clusters

High-availability clusters (High-Availability Data Replication, shared disk secondary servers, and remote stand-alone secondary servers) provide a way to provide one or more copies of the database server. LBAC objects created on a database of the primary server are replicated to the secondary servers. All tables protected on the primary server are protected on the secondary servers.

Distributed queries

You can query more than one database on the same database server or across multiple database servers. This type of query is called a distributed query. LBAC rules are applied to distributed queries involving protected tables and local synonyms of remote protected tables. Queries issued from a non-LBAC server but involving LBAC-protected tables on a different server also require that the user have the necessary LBAC credentials to access the protected data on the other server.

Fragmentation

You can use fragmentation to control where data is stored at the table level using a fragmentation strategy. IBM Informix ensures that the source and targets tables have the required identical LBAC security objects for attaching and detaching fragments.

Synonyms and views

Views and synonyms can be created on existing tables and views that are located in the current database, or in another database of the local database server or of a

remote database server. LBAC rules are applied when a user attempts to access data through views and synonyms on protected tables.

Violations tables

IBM Informix provides a facility to track rows that violate constraints. The START VIOLATIONS TABLE statement creates a special violations table that holds nonconforming rows that fail to satisfy constraints and unique indexes during INSERT, UPDATE, and DELETE operations on target tables. In order to prevent unauthorized exposure of protected data through a violations table, IBM Informix secures the violation table with same security policy as the target table when the START VIOLATIONS TABLE statement is executed.

Referential integrity scans

LBAC rules are applied when the ON DELETE CASCADE option is specified and when an INSERT statement to a child table generates a referential integrity scan on the parent table.

Part 2. Auditing data security

This section contains information about how to audit the security of your database.

Chapter 7. Overview of auditing

This chapter provides an overview of auditing and of 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 introduces the various audit administration roles.

Secure-auditing facility

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

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

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

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

Audit events

Any database server activity that can potentially alter or reveal data or the auditing configuration is considered an *event*. You can use the database server secure-auditing facility to audit and keep a record of events either when they succeed or fail, or when the activity is attempted. You can identify each audit event by a four-letter event code. Chapter 12, "Audit event codes and fields," on page 12-1 lists the audit-event codes and describes the events that you can audit with the secure-auditing facility.

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

Audit masks

Audit masks specify those events that the database server must 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 must specify which audit events to audit. To specify audited events, add the events to the masks. You must also perform other tasks, which Chapter 8, "Audit administration," on page 8-1, describes.

The database server does not provide auditing for objects or processes. For example, you cannot ask the database server to audit all access attempts on a certain object. You can, however, filter audit records from the audit trail based on objects with the audit-analysis tools, which Chapter 9, "Audit analysis," on page 9-1, describes.

Figure 7-1 represents a set of audit masks. The actual masks and their features are explained in "Audit masks and audit instructions" on page 7-5.

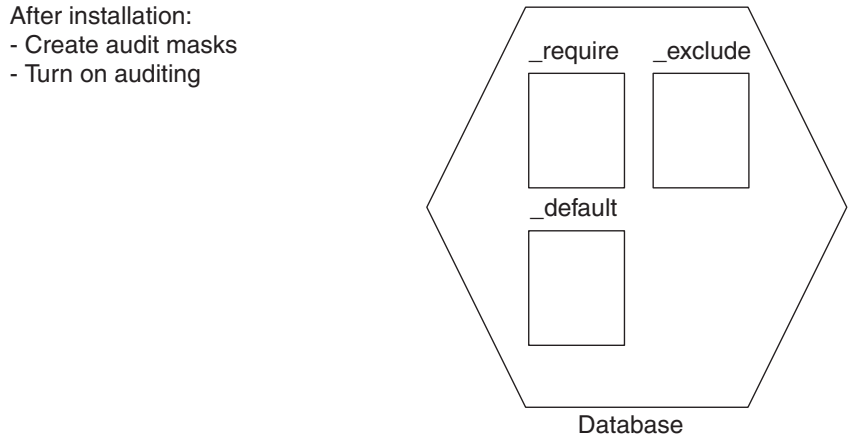


Figure 7-1. Audit masks after installation

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

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

In addition to the three masks that Figure 7-1 shows, you can specify user masks for individual users. You can use user masks to audit some users more than others and target different types of activities for different users. Except for the audit administrator who maintains the masks, a user cannot tell which events are being audited. For a description of user masks, see "User masks" on page 7-6.

You can also create template masks for creating new user masks. For a description of template masks, see "Template masks" on page 7-7.

Masks and their events are called auditing instructions, as Figure 7-2 on page 7-3 shows. You have significant flexibility regarding the auditable facets of Informix. You can select anything from minimal audit instructions, in which no events are audited, to maximal audit instructions, in which all security-relevant database server events are audited for all users.

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

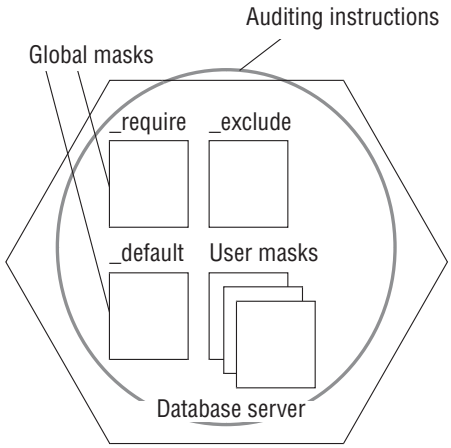


Figure 7-2. The auditing instructions

After you define the auditing instructions and turn on auditing, you can modify one or more audit masks as requirements change and you identify potential security threats. For information about how to change audit masks, see Chapter 8, "Audit administration," on page 8-1.

Related reference:

"The onaudit utility: Configure audit masks" on page 10-1

Selective row-level auditing

Auditing can be configured so that row-level events of only selected tables are recorded in the audit trail. Selective row-level auditing can compact audit records so that they are more manageable and potentially improve database server performance.

The **onaudit** utility supports an option (the `-R` flag) that can be run to enable selective row-level auditing. The CREATE TABLE and ALTER TABLE statements are used as SQL commands that flag specific tables for inclusion in the row-level audit event records.

You can start selective row-level auditing either when you initially start auditing of your databases or while the auditing utility is already running.

One reason to use selective row-level auditing is that it can filter out auditable events that are not important to database security. For example, an administrative user of an Informix installation with confidential data must be able to track when users perform actions on the database server that endanger the security of the system. With row-level auditing of all tables on the system, the audit record contains information about auditable events on system tables that contain reference information for database administration and tables that contain sensitive confidential information. If the administrator must investigate a security breach by examining the audit records, there can be large amounts of information from the system tables that hinder finding the relevant event on the tables containing the confidential data. By flagging only the security-critical tables for row-level auditing, the audit trail is parsed to a more compact set of records that is easier to analyze.

Related tasks:

"Setting up selective row-level auditing" on page 8-8

Audit process

When you turn on auditing, the database server generates audit records for every event that the auditing instructions specify, as Figure 7-3 shows.

The database server stores the audit records in a file called an audit file, as Figure 7-3 shows. The collection of audit records makes up the audit trail. (The audit trail might consist of more than one audit file.)

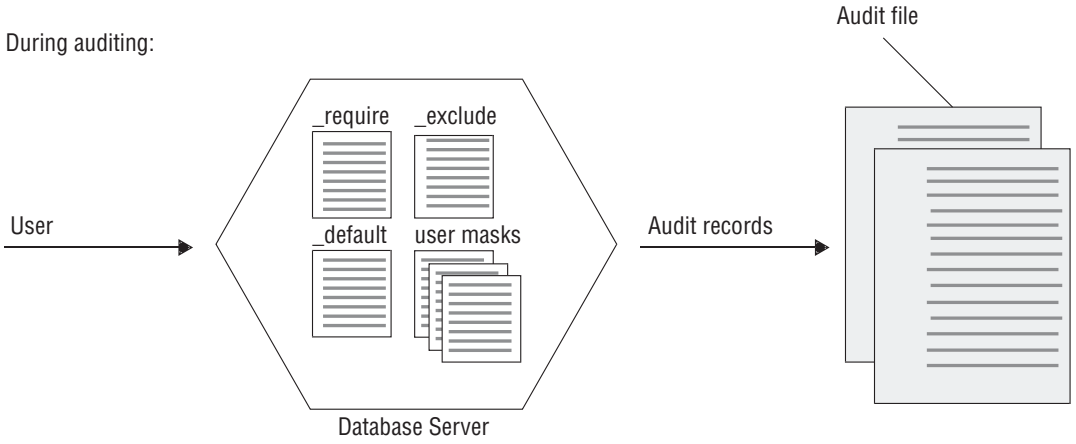


Figure 7-3. The audit process

An audit administrator must specify and maintain the audit configuration, which includes the following information:

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

These topics are explained in "Audit configuration" on page 7-9.

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

In high availability clusters, only the primary database server performs secure auditing and produces an audit trail. The **onaudit** utility runs on the secondary servers but does not audit any of the audit events.

Audit trail

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

After you extract data, you can specify that it be formatted to load into a database for subsequent manipulation with SQL. "Audit analysis overview" on page 7-15 explains this process.

When the database server stops writing to one audit file and begins writing to a different audit file, an event alarm is generated. If you use an alarm program, you can modify it to watch for the new audit event to archive audit records, monitor records, or remove them. See the event alarms documentation in *IBM Informix Administrator's Reference* for more information about how to make use of the audit event notification.

Details about the Audit Trail Switch Event Alarm:

- Class ID: 72
- Severity: 3
- Class Message: Audit trail is switched to a new file
- Message: This message is displayed when the database server switches to a new audit trail file.

Roles for database server and audit administration

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

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

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

For detailed information about roles and role separation, see "Using role separation" on page 8-3. For information about setting up role separation and creating a user group for each role, see your *IBM Informix Installation Guide*.

Audit masks and audit instructions

As described in "Audit masks" on page 7-1, an audit mask specifies a set of events to be audited when a user performs them. Audit events are derived from a combination of user and global masks. Chapter 12, "Audit event codes and fields," on page 12-1 lists the set of auditable events. The set of events is fixed, but you use masks to specify only the ones that you are required to audit.

The following table lists four types of audit masks.

Mask Type
Mask Name

Individual user masks
username

Default mask
_default

Global masks
_require and _exclude

Template masks
_maskname

The following section describes the first three kinds of masks. For a description of template masks, see page "Template masks" on page 7-7.

User masks

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

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

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

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

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

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

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

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

Template masks

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

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

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

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

- _default
- _require
- _exclude

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

Audit instructions

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

This section suggests how to choose events to audit, how to set the audit instructions, and how the choices affect performance. For details of how to create and modify audit masks, see Chapter 8, "Audit administration," on page 8-1.

All the audit masks that the database server uses are stored in the system-monitoring interface (SMI) sysaudit table in the sysmaster database. The masks are updated automatically when the database server is upgraded to a newer version. Although information stored in the sysmaster database is available through SQL, you must use the **onaudit** utility for all audit-mask creation and maintenance. (See "The onaudit utility: Configure audit masks" on page 10-1.) Also, see the description of the sysmaster database in the *IBM Informix Administrator's Reference*.

Resource and performance implications

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

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

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

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

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

When you are configuring the auditing parameters for your system, determine what actions the database server takes if it becomes unable to write to audit files, such as when the audit trail exceeds available storage capacity.

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

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

Special auditing considerations

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

Level of auditing granularity

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

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

- Read Row (RDRW). The partition and the record ID within the partition are in the event record.
- Update Row (UPRW). The partition and the record ID within the partition are in the event record.

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

For more information about the fields in an audit-event record, see Chapter 12, "Audit event codes and fields," on page 12-1.

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 about the corresponding SQL statements, see the *IBM Informix Guide to SQL: Syntax*.

Use of various masks

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

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

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

Audit configuration

The AAO can monitor the audit configuration, as Chapter 8, "Audit administration," on page 8-1 describes. Setting the audit configuration consists of performing the following tasks:

- Turning auditing on or off
- Specifying audit modes
- Using the `ADTCFG` file
- On UNIX, determining properties of the audit files

Sections that follow describe these topics.

Auditing on or off

An audit administrator determines whether auditing is on or off. Auditing is turned off by default when the database server is installed. As Chapter 8, "Audit administration," on page 8-1, describes, the AAO can turn auditing on and off at any time, by using the **onaudit** utility, which "The onaudit utility: Configure audit masks" on page 10-1, describes. The database server can be in either online or quiescent mode for the changes to take effect.

When the AAO turns on auditing, all sessions, new and current, start auditing auditable events. Both existing sessions and new sessions produce records. All user sessions that are started thereafter also produce audit records.

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

The ADTCFG file

Configuration parameters in the ADTCFG file specify the properties of the audit configuration. These configuration parameters are **ADTERR**, **ADTMODE**, **ADTPATH**, and **ADTSIZE**.

The path name for the default ADTCFG file follows.

UNIX $INFORMIXDIR/aaodir/adtcfg

Windows

%INFORMIXDIR%\aaodir\adtcfg

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

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

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

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

Changes made with **onaudit** are written to an adtcfg.*servernum* companion file. (**SERVERNUM** is a parameter in the onconfig file, which the *IBM Informix Administrator's Reference* describes.) The configuration changes take effect in the server immediately. The current and subsequent server instance uses the adtcfg.*servernum* file for the audit configuration parameters instead of the file adtcfg.

For details, see “The onaudit utility: Configure auditing” on page 10-5 and see Chapter 13, “The ADTCFG file,” on page 13-1. For more information about auditing administration, see “Administrative roles and role separation” on page 8-1.

Properties of audit files

As “Audit process” on page 7-4 describes, with database server-managed auditing , the database server writes audit records to audit files in an audit trail. This section describes the audit files in more detail.

Location of audit files (UNIX)

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

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

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

Location of audit files (Windows)

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

Applications cannot write to the Windows Security Event log, so auditing messages from the database server are now sent to a log file, whose directory path can be specified by using the **onaudit** utility. The default path name is `%INFORMIXDIR%\aaodir`.

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

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

New audit files

The database server creates a new audit file under the following conditions:

- When you initialize the database server
- When you restart the database server after being offline
- When the file reaches a specified size
- When you manually direct the database server to start a new audit file
- When you start database server-managed auditing

When the database server writes an audit record, the database server appends the record to the current audit file. If the database server goes offline and is restarted, it starts a new audit file. The ADTLOG file,`$INFORMIXDIR/aaodir/adtlog.`*`server`*,

maintains the number of the audit log currently being used. The number in the ADTLOG file increases by one each time the server restarts, and is used as a starting point when the server checks for and numbers new log files. The server still checks if the file with the name dbservername.*number* already exists in the directory. If the database server detects an existing file, the audit facility does not modify it. The number is increased and the process is repeated until an unused number is found, and the skipped files are reported in the online log file. Informix creates the ADTLOG file if it does not exist.

Audit file names

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

The naming convention is dbservername.*integer*, where *dbservername* is the database server name as defined in the onconfig file, and *integer* is the next available integer after the number defined in the ADTLOG file. Each server's audit file series starts with 0.

For example, if a new audit file is started for a database server maple, and the last audit record was saved in the file maple.123, then the next audit file is named maple.124. If maple.124 already exists, the next available number is used. The names are unique to a specific audit directory, so both auditdir1/maple.123 and auditdir2/maple.123 are acceptable, but writing to a new directory does not change the file checking and naming that begins with the number in the ADTLOG.

Audit file numbers do not repeat unless you remove audit log files and delete the ADTLOG file.

Windows Message Server

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

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

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

Error modes for writing to an audit file

If the database server encounters an error when it writes to the audit file, it can behave in various ways called error modes. You can change the error mode, as "Setting the error mode" on page 8-6 describes, at any time during database server operation, even after an error occurs. See also the explanation of **onaudit** error modes in "The onaudit utility: Configure audit masks" on page 10-1.

Halt error modes

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

- Mode 1: A thread is suspended but the session continues when the audit record is successfully written.

- Mode 3: The database server shuts down and the user session cannot continue.

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

Continue error mode

When the database server is in continue error mode (0), it allows the session that received the error to continue processing after it writes to the audit trail. However, the audit record that was being written when the error occurred is 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.

Access to the audit trail

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

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

Access to audit files on UNIX

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

Ownership:
informix

Group ID:
same as `$INFORMIXDIR/aaodir`

Permissions:
775

Important: The AAO must 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`) does not have adequate protection.

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

`aaodir`

Ownership:
informix

Group ID:
informix

Permissions:
775

`aaodir/adtcfg.std`

Ownership:
informix

Group ID:
informix

Permissions:
644

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

`aaodir`

Ownership:
informix

Group ID:
<aao_group>

Permissions:
775

`aaodir/adtcfg.std`

Ownership:
informix

Group ID:
<aao_group>

Permissions:
644

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

Access to audit records on Windows

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

Ownership:
informix

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

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

`aaodir`

Ownership:
informix

Group ID:
Administrator

`aaodir\adtcfg.std`

Ownership:
database server administrator

Group ID:
Administrator

Audit analysis overview

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

Importance of audit analysis

The database server audit mechanism is designed to both deter and reveal attempted and 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, however, you routinely analyze and review the audit data, you might discover suspicious activity before a successful violation occurs. The first step to terminate any security violation is to detect the problem. If a database server violation occurs, you can use the audit trail to reconstruct the events that lead up to and include this violation.

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

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

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

Periodically check the types of events that are audited versus the data that actually is 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. After you identify a potential security anomaly, you can then investigate further to determine if anyone on the database server attempts to obtain unauthorized information, if a user misuses the database server, or if a user becomes lenient in self-regulated security enforcement.

Preparation for audit analysis

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

- The first method displays audit data as it appears in the audit trail, which you can subject to your own audit-analysis tools. This method guarantees accuracy because no processing is done on the raw audit records.
- The second method converts the audit records into a form that can be uploaded into a table that the database server manages. You can then use SQL to generate reports based on this data. With the SQL-based method, you can create and use customized forms and reports to manipulate and selectively view audit data,

which provides a flexible and powerful audit-analysis procedure. Be sure, however, that records are not deleted or modified from either the intermediate file or from the database before 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 9, "Audit analysis," on page 9-1 and "The onaudit utility: Configure audit masks" on page 10-1 describe. For either method, you can extract audit events for specific users, database servers, or both.

To perform audit analysis, first have audit records in your database server. 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.

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

Strategies for audit analysis

The primary threat to database server security is unauthorized disclosure or modification of sensitive information. This section contains information about 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 can indicate, for example, that a user is attempting to give sensitive data to another user who does not have the correct UNIX permissions or Windows access privileges to access the data.

Event success

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

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

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

Insider attack

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

Browsing

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

Aggregation

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

Responses to identified security problems

After you identify the user or users who are responsible for irregularities in the security audit trail, see 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 contains information about responses to various kinds of security threats to the DBMS. For more information about various roles, see "Administrative roles and role separation" on page 8-1.

Primary threats

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

Users must belong to the correct group to access the database server. They must also have a valid login ID in the operating-system password file.

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

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

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

Privileged activity threats

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

Database Server Administrator

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

Database System Security Officer

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

Operating-System Administrator

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

Audit Analysis Officer

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

Shared-memory connection threats on UNIX

A shared-memory connection provides fast access to a database server if the client and the server are on the same computer, but it poses some security risks. False or nontrusted applications can delete 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 default path name for the UNIX configuration file is `$INFORMIXDIR/etc/sqlhosts`.

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

Threats from malicious software

Database users can easily and unknowingly download malicious or unauthorized software. This is a security threat that can come from not only server machines that host the databases, but also computers used to access the databases.

To protect the database server from malicious software:

- Keep the database server on a different computer from the clients that must connect to it
- Restrict access to the computer hosting the database server
- Monitor the software installed on the database server computers (for example, by running a checksum process periodically)
- Keep a record of all the files and permissions on the database server computer
- Institute a strict security policy
- Make all users aware of the dangers of starting software of unknown or untrusted origin

Malicious software can defeat security controls in many ways. For example, such software can copy data for subsequent access by an unauthorized user or grant database access privileges to an unauthorized user.

Remote-access threats

When a user is granted database 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 might 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 data access 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 access privileges of the software or the individual user audit mask associated with that user name are not removed before the same user name is allocated to a new user, the new user inadvertently inherits the privileges and audit mask of the previous user.

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

Untrusted software used in a privileged environment

Problems might occur if DBSAs or OSAs run untrusted software. Untrusted software can use the privileges of the DBSA or the OSA to perform actions that bypass or disable the security features of the product or that grant inappropriate access privileges.

The primary countermeasure to this vulnerability is to make sure that DBSAs and OSAs do not run software of unknown or untrusted origin. Operating-system access controls must be used to protect all software that DBSAs and OSAs run against unauthorized modification.

Distributed database configuration threats

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

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

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

Chapter 8. Audit administration

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

Administrative roles and role separation

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

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

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

Database Server Administrator

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

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

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

If the DBSA group is not group **informix**, the permissions on **oninit** must be modified to 6755 (granting others execute permission) so that members of the new DBSA group can start the database server

The DBSA is responsible for granting or revoking the EXTEND role to restrict users who can register DataBlade modules or external user-defined routines (UDRs).

Database System Security Officer

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

These tasks include the following actions:

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

The DBSSO performs these tasks with the **onaudit** utility. For information, see Chapter 10, "The onaudit utility," on page 10-1.

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

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

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

Audit Analysis Officer

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

For database server-managed auditing on UNIX, the AAO also determines the directory for the audit trail and the maximum size of each audit file.

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

The AAO performs these tasks with the **onaudit** and **onshowaudit** utilities, which "The onaudit utility: Configure audit masks" on page 10-1 describes. If the AAO uses **onaudit** to change the audit configuration parameters during a database server session, the new values are written to the `adtcfg.`*servernum* file for that instance of the database server.

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

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

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

Other administrative roles and users

A number of other, more minor, roles might be involved in database server secure auditing.

Database Administrator

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

Operating-System Administrator

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

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

System Users

All operating-system accounts, including those for the DBSA, DBSSO, AAO, and the account called **informix**, potentially can use the database server. All users with accounts who want to use the database server must explicitly be granted access to the database server if role separation is configured to enforce access control on database server users. The DBSA can revoke that access at any time, whether role separation is enabled. For more information about granting or revoking access, see "Configuring and enforcing role separation" on page 8-4.

Privileged Users

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

Using 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) must be different from the person who monitors the audit trail (AAO), and both must be different from the person who is responsible for the operations of the database server (the DBSA).

Assigning roles

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

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

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

- Have as few DBSA and DBSSO accounts as possible.

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

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

  The AAO determines whether to audit all DBSA or DBSSO actions in the system. It is essential that someone with a role different from that of the DBSA or DBSSO be in charge of auditing configuration, so that all users, including the DBSA and DBSSO, are held accountable for their actions in the system. This constrains users to perform only those tasks that their duties specify and limits the risk of compromising security.
- Limit access to the account **informix** because it can bypass role-separation enforcement and other database server access-control mechanisms.

Configuring and enforcing role separation

The DBSA, or the person who installs the database server, enforces role separation and decides which users are the DBSSO and AAO. To find the group for the DBSA, DBSSO, or AAO, look at the appropriate subdirectory of `$INFORMIXDIR` on UNIX or `%INFORMIXDIR%` on Windows.

On Windows, role separation is configured only during installation. On UNIX, you normally configure role separation during installation, but you can also configure it after the installation is complete or after the database server is configured. The OSA who installs the software enforces role separation, and decides which users (Windows) or groups (UNIX) are the DBSSO and AAO. On UNIX, the group that owns `$INFORMIXDIR/aaodir` is the AAO group; the group that owns `$INFORMIXDIR/dbssodir` is the DBSSO group. By default, group **informix** is the DBSSO, AAO, and DBSA group.

On UNIX, if you use the InstallShield MultiPlatform (ISMP) installer in GUI or terminal mode to install the database software, you are asked if you want to configure role separation. If instead you use the scripted bundle installer, then the environment variable INF_ROLE_SEP controls whether you are asked to set up separate roles. If the INF_ROLE_SEP environment variable exists (with or without a value) role separation is enabled and you are asked to specify the DBSSO and AAO groups. (You are not asked about the DBSA group.) If the INF_ROLE_SEP environment variable is not set, then the default group **informix** is used for all these roles.

You are not required 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.

After the installation is complete, INF_ROLE_SEP has no effect. You can establish role separation manually by changing the group that owns the `aaodir`, `dbssodir`, or `etc` directories. You can disable role separation by resetting the group that owns

these directories to **informix**. You can have role separation enabled for the AAO without having role separation enabled for the DBSSO.

Role separation control is through the following group memberships:

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

Note: For each of the groups, the default group is the group **informix**.

The **ls -lg** UNIX command produces the following output showing role separation:

```
total 14
drwxrwx---  2 informix       ix_aao    512 Nov 21 09:56 aaodir/
drwxr-xr-x  2 informix       informix 1536 Nov 30 18:35 bin/
drwxrwx---  2 informix       ix_dbsso  512 Nov 30 10:54 dbssodir/
drwxr-xr-x 10 informix       informix  512 Nov 21 09:55 demo/
drwxrwxr-x  2 informix       informix 1024 Nov 30 11:37 etc/
```

In the preceding example, the AAO belongs to the group **ix_aao**, the DBSSO belongs to the group **ix_dbsso**, and the DBSA belongs to the group **informix**.

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

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

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

Auditing setup

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

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

Setting up the default and global masks

Before setting up default and global masks, the DBSSO must understand how the various masks work and what the implications are for different auditing instructions. Also, the DBSSO must understand which auditing events to place in which masks. For details, see Chapter 7, "Overview of auditing," on page 7-1.

Use the **onaudit** utility to add audit events to audit masks. Chapter 12, "Audit event codes and fields," on page 12-1 lists the audit events and their codes. "The onaudit utility: Configure audit masks" on page 10-1 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:

```
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 "The onaudit utility: Configure audit masks" on page 10-1.

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 directory for the audit trail (UNIX)

The database server stores audit files in a file system directory. You can specify the directory with the **onaudit** utility. For example, the following command specifies `/work/audit` as the UNIX file system in which the database server is to store audit files:

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

Note: The `onaudit -p /work/audit` command works only if logging is enabled or if `-l N` options are included in the command line.

You can change the audit directory at any time. You can also set up the type of auditing and specify the directory with the `ADTCFG` file, which is described in Chapter 13, "The ADTCFG file," on page 13-1.

For more information about the **onaudit** utility, see "The onaudit utility: Configure audit masks" on page 10-1.

Related reference:

"The onaudit utility: Configure auditing" on page 10-5

Setting the error mode

As Chapter 7, "Overview of auditing," on page 7-1 describes, the database server has three actions that it can perform if an error occurs when writing to the audit trail: a continue-error mode, and two levels of severity of halt-error mode. Be sure that you 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 "The onaudit utility: Configure audit masks" on page 10-1. For the **ADTERR** configuration parameter, see Chapter 13, "The ADTCFG file," on page 13-1.

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
```

Related reference:

"The onaudit utility: Configure auditing" on page 10-5

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.

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

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

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

You can reconfigure auditing as usage changes and potential security threats are identified. For the **onaudit** syntax, see "The onaudit utility: Configure audit masks" on page 10-1. For information about the **ADTMODE** configuration parameter, see Chapter 13, "The ADTCFG file," on page 13-1.

Important: Although database server audit-record generation might have a negative effect on database server performance and resources, you must perform more than the minimal database server audit. This additional audit improves the likelihood of detecting security violations and 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 must enable full auditing of the suspect user to get a complete picture of the user's activities.

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

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

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

For more information about database server performance considerations, see your *IBM Informix Performance Guide*.

Related reference:

"The onaudit utility: Configure auditing" on page 10-5

Setting up selective row-level auditing

Auditing can be configured so that row-level events of only selected tables are recorded in the audit trail. Selective row-level auditing can compact audit records so that they are more manageable and potentially improve database server performance.

You must be a DBSSO to complete this task.

1. Run the **onaudit** command with the `-R` option.
2. Designate the tables that you want to audit on the row level:
   a. For each existing table that you want to audit at the row level, run the ALTER TABLE statement with the ADD AUDIT clause.
   b. For each new table that you want to audit at the row level, run the CREATE TABLE statement with the WITH AUDIT clause.

The following code examples and descriptions illustrate how to enable selective row-level auditing.

The `onaudit -R 1` command enables selective row-level auditing, and the `onaudit -c` command displays the audit configuration for verification. The audit configuration information indicates that the **ADTROWS** parameter is correctly set to 1.

```
$ onaudit -R 1
$ onaudit -c
Onaudit -- Audit Subsystem Configuration Utility

Current audit system configuration:
    ADTMODE    = 1
    ADTERR     = 0
    ADTPATH    = /usr2/support/chunks/IDS1170FC1B1
    ADTSIZE    = 50000
    Audit file = 0
    ADTROWS    = 1
```

The `onaudit -a -u _default -e +DLRW,INRW,RDRW,UPRW` command creates the user audit mask `_default` and sets the granularity to **Delete Row**, **Insert Row**, **Read Row**, and **Update Row** audit events. The `onaudit -o -y` command displays the audit mask for verification.

```
$ onaudit -a -u _default -e +DLRW,INRW,RDRW,UPRW
$ onaudit -o -y
Onaudit -- Audit Subsystem Configuration Utility

_default                                -       DLRW,INRW,RDRW,UPRW
```

In the following part, the table `state` is flagged for selective row-level auditing and values are inserted to test whether the action is captured in the audit records.

```
$ dbaccess stores_demo -
Database selected.
> ALTER TABLE state ADD AUDIT;
Table altered.

> INSERT INTO state VALUES ('FR', 'France');

1 row(s) inserted.
```

Finally, the onshowaudit command is run to display the audit record. The results indicate that selective row-level auditing is functioning.

```
$ onshowaudit

ONSHOWAUDIT Secure Audit Utility
INFORMIX-SQL Version 11.70.FC1
ONLN|2010-11-01 15:23:24.000|fido|765|abc1170shm|
informix|0:INRW:stores_demo:106:1048999:309::

Program Over.
```

Related concepts:

"Selective row-level auditing" on page 7-3

Using the WITH AUDIT Clause (SQL Syntax)

Related reference:

"The onaudit utility: Configure auditing" on page 10-5

ADD AUDIT Clause (SQL Syntax)

DROP AUDIT Clause (SQL Syntax)

Activating auditing

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

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

```
onaudit -l 1
```

After you turn on auditing, it takes effect immediately for all sessions.

The AAO can configure the database server to turn on auditing when the server starts when the **ADTMODE** configuration parameter is set to the numbers 1, 3, 5, or 7 in the ADTCFG file. For details on **ADTMODE** parameter values, see "The onaudit utility: Configure auditing" on page 10-5 and Chapter 13, "The ADTCFG file," on page 13-1.

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

To turn off auditing after it starts, see "Turning off auditing" on page 8-15.

Related reference:

"The onaudit utility: Configure auditing" on page 10-5

Audit mask maintenance

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

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

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

Creating audit masks

You can create masks that more closely match the types of activities that individual users perform than do default and global masks.

- To create individual user masks, specify user IDs as mask names.
- To create template masks, preface the name of a mask with an underscore (_). Chapter 7, "Overview of auditing," on page 7-1 describes template masks and user masks.

You specify events in the mask when you create it by using the audit events from the alphabetic listing in the table Chapter 12, "Audit event codes and fields," on page 12-1. 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 require just two templates for familiar and unfamiliar users that you copy to individual user masks: _guest and _trusted.

Creating a template mask

To create a template audit mask:

Use the **onaudit** utility. The "The onaudit utility: Configure audit masks" on page 10-1 shows the syntax. The following example shows how to create a template mask called _guest with the audit events Create Database, Grant Database Access, and Grant Table Access:

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

Creating a user mask from a template mask

A mask that is used as the foundation for one or more other masks is a base mask.

To create a user mask from a template mask:

Create the template mask. After you create a template mask for a given user category, you can use it as the basis of masks for individual users, adding or removing only the audit events that differ for each user.
The following example creates a user mask for the user `terry`, based on the `_guest` template mask:

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

The `terry` mask has the same audit events as the `_guest` mask, except for the `CRDB` (Create Database) audit event, which was removed.
Instead of template masks, you can also use existing user _default, _require, and _exclude masks as base masks.

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

Creating a user mask without a template mask

To create user masks without a template mask:

Use events as the basis for the user 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 "The onaudit utility: Configure audit masks" on page 10-1.

Adding one or more masks using an input file

To add one or more masks by using an input file:

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
```

The following code block shows an example of an input file. The syntax for the input file is explained in "The onaudit utility: Configure audit masks" on page 10-1.

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

The preceding example input file provides the following information:

- In the first line, the instructions specify auditing for user `kickt` in the new template `_secure1`.
- The second line creates a new mask called `jacks`, which contains the events Add Chunk (`ADCK`), successful attempts at Read Row (`SRDRW`), and all attempts at Grant Database Access (`GRDB`) and Open Database (`OPDB`).
- In the third line, the user `pat` is audited for all events that are specified in the template `_secure2`, and also for all attempts at Alter Table (`ALTB`), but not for attempts at Create Table (`CRTB`), Create Index (`CRIX`), and Start New Session (`STSN`).
- No template is specified for the target mask `jaym` in the fourth line, and no events are indicated; the mask is empty. (This prevents the _default mask from being applied to `jaym`.)
- In the fifth line, the target mask `johns` audits the same events as the mask `akee`, minus all successful attempts at Alter Index (`SALIX`).

Important: Future changes to a base mask are not reflected in other masks that might have been created or modified with that mask as a base.
An example of an audit mask input file, `adtmasks.std`, is provided in the `$INFORMIXDIR/aaodir` UNIX directory or in the `%INFORMIXDIR%\aaodir` Windows directory. The `adtmasks.std` file is intended only to serve as a guide to the DBSSO for how to set up an audit mask.
Audit masks do not work the same way as audit configuration parameters during

initialization of the database server. (See “The ADTCFG file” on page 7-10.) Specifically, audit masks are not automatically read from a file and initialized.

Related reference:

“The onaudit utility: Configure audit masks” on page 10-1

Displaying audit masks

To display all the audit masks and the audit events that each mask contains:

Use the `-o` option of the **onaudit** utility. When you issue the **onaudit -o -y** command, the output (mask name, base mask, audit events) are displayed as follows:

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

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

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

A list of audit masks is helpful when you must modify them. You can use the modified output as an input file to modify a single mask or groups of masks in a single batch. For more information, see “Modifying audit masks.” For the complete syntax of the **onaudit -o** option and a description of the output, see “The onaudit utility: Configure audit masks” on page 10-1.

Tip: If you use a base mask to create or modify a mask, the base mask itself is not displayed 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.

To modify audit masks:

Use the `-m` option of the **onaudit** utility to modify a single mask. You can use this option to use another mask as a base to add or remove individual audit events. 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.
The following example shows how to modify the user mask `pat`. The `_guest` template mask forms a base from which a complete set of audit events is drawn. Settings for specific events from that file are then superseded by the events listed as arguments to the `-e` option.

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

When you supply a base mask with the `-r` option, you replace all the audit events in the initial mask. When you change only a few events in a mask, you might not want to specify a base mask. For the syntax and another example of how to modify a mask, see “The onaudit utility: Configure audit masks” on page 10-1.

Deleting audit masks

To delete a single mask or all masks:

Use the `-d` option of the **onaudit** utility. The following example deletes the individual user mask for user `terry`:

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

For the syntax of the **onaudit** utility, see "The onaudit utility: Configure audit masks" on page 10-1.

Audit configuration maintenance

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

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

This section describes how to use **onaudit** to perform these tasks. For the syntax of the **onaudit** utility, see "The onaudit utility: Configure audit masks" on page 10-1.

Displaying the audit configuration

To display the current audit configuration use the `-c` option of the **onaudit** utility.

- On UNIX, the following example shows output from the `onaudit -c` command.

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright (c) IBM Corp., 1998 - 2010

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

  In the preceding example, 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.
  - **ADTROWS** is set to 0, which indicates that selective row-level auditing is turned off.

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

- On Windows, the example shows output from the `onaudit -c` command.

```
onaudit -c

Onaudit -- Audit Subsystem Control Utility
Copyright IBM Corporation 1996, 2010 All rights reserved

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

  In the preceding example, the current audit system is configured as follows:

  - **ADTMODE** is set to 1, which indicates that database server-managed auditing is on.
  - **ADTERR** is set to 0, which indicates a continue error mode.
  - **ADTPATH** shows the default directory for audit files.
  - **ADTSIZE**, which represents the maximum size of the audit file, is specified as 50,000 bytes.
  - The number of the current audit file in the current audit directory is 0, meaning that no other audit file exists in the current series.
  - **ADTROWS** is set to 0, which indicates that selective row-level auditing is turned off.

Related reference:

"The onaudit utility: Configure auditing" on page 10-5

Starting a new audit file

You use the **onaudit** command to start a new audit file. For the **onaudit** syntax to start a new audit file, change the audit-file size, or change the path name of the audit directory, see Chapter 10, "The onaudit utility," on page 10-1.

You can use more than one flag at a time in an **onaudit** command.

You can start a new audit file in one of the following ways:

- Use `onaudit -s` to change the maximum size of an audit file. If the audit file is already larger than the new size that you specify, the utility saves the current file and starts to write to a new one.

  The following example changes the default size to 20,000 bytes:

  ```
  onaudit -s 20000
  ```

- Use `onaudit -n` to start a new audit file without changing the maximum size.

  This option, which the following example shows, saves the current audit log to another file whenever you run it:

  ```
  onaudit -n
  ```

- Use `onaudit -p` to change the directory in which the database server writes audit files.

  The following example specifies `/work/audit` as the UNIX file system where the audit files are to be kept:

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

The directory that you specify must exist.

- Start database-server- managed auditing. A new audit file starts every time that you start database-server- managed auditing.

Related reference:

"The onaudit utility: Configure auditing" on page 10-5

Changing audit levels

- Use the **onaudit** utility to change levels of auditing by the database server and to change the mandatory auditing of the DBSA.

  For example, to start basic auditing, enter the following command:

  ```
  onaudit -l 1
  ```

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

  ```
  onaudit -l 5
  ```

Related reference:

"The onaudit utility: Configure auditing" on page 10-5

Changing the audit error mode

As Chapter 7, "Overview of auditing," on page 7-1 and "Setting the error mode" on page 8-6 explain, the database server behaves in one of three ways if it encounters an error when it writes to the current audit file.

To change the audit error mode:

Use 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
```

Related reference:

"The onaudit utility: Configure auditing" on page 10-5

Turning off auditing

To turn off auditing:

Use the **onaudit** utility.
The following example shows the command that turns off auditing:

```
onaudit -l 0
```

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

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

Related reference:

"The onaudit utility: Configure auditing" on page 10-5

Chapter 9. Audit analysis

The audit analysis is extremely important. This chapter contains the following information:

- 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 7, "Overview of auditing," on page 7-1.

Audit-record format

The database server generates the second part of the audit record, with fields that depend on the audit event.

Table 9-1 shows the format of the database server audit records.

Table 9-1. Audit-record format

ONLN	date and time	hostname or hostname. domain.ext	pid	database server name	user name	errno	event mnemonic	Additional Fields
ONLN	2008-07-28 15:43:00.000	turk	4549	khan	jazt	0	CRDB	dbsch
ONLN	2008-07-28 15:43:18.000	turk	4549	khan	jazt	0	ACTB	dbsch:jazt:v1:103
ONLN	2008-07-28 15:43:19.000	turk	4549	khan	jazt	0	CLDB	dbsh
ONLN	2008-07-28 15:43:21.000	turk	4549	khan	jazt	0	ALFR	local:109:-:-:4:4: db1,db2,db3, rootdbs:0
ONLN	2008-07-28 15:43:28.000	turk	4549	khan	jazt	0	ALFR	local:109:aa5x:-: 32:4: db1,db2 rootdbs:0
ONLN	2008-07-28 15:43:29.000	turk	4549	khan	jazt	0	STDS	2:-
ONLN	2008-07-28 15:43:29.000	turk	4549	khan	jazt	0	STPR	100
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

ONLN
A fixed field used to identify events

date and time
Indicates when the audit event was recorded

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

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

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

database server name
The name of the database server on which the audit event is run

user name
The login name of the user who requests the event

errno The event result that contains the error number that the event returns, indicating success (0) or failure

event mnemonic
Database server audit event that the database server ran, such as ALFR (Alter Fragment)

additional fields
Any fields that identify databases, tables, and so on. These additional fields are audit-event fields that contain information captured in tabular form by the **onshowaudit** utility for audit analysis.

For operating-system-managed auditing on UNIX, the database server audit record is an additional field for the operating-system audit record. Chapter 12, "Audit event codes and fields," on page 12-1 lists the audit-event fields.

Audit analysis without SQL

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

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

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

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

The command-line syntax for how to extract information with **onshowaudit** is explained in "The onaudit utility: Configure audit masks" on page 10-1.

Audit analysis with SQL

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

Planning for SQL audit analysis

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

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

Whether you perform analysis with DB-Access or build a customized application, remember the advice given for audit review in "Audit analysis overview" on page 7-15. To view audit events for specific objects, select rows based on their value in the dbname, tabid, or row_num column.

If you discover suspicious activity based on initial analysis of the audit table in the database server, you might increase the scope of your collection of audit events to pinpoint the problem. If you feel certain you have a security problem, see "DBMS security threats" on page 7-18.

Revoking and granting privileges to protect audit data

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

Tables that you create in non-ANSI compliant databases have privileges that allow all users access. Although the default database permissions or access privileges prevent access to the tables, correct 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 about SQL statements, see the *IBM Informix Guide to SQL: Syntax*.

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

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

Preparing audit analysis records for SQL access

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

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

Creating a data file for dbload

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

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

Important: You must remove the six header lines that are 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 "The onaudit utility: Configure audit masks" on page 10-1.

Creating a database for audit data

To load data files into a database with **dbload**, a database to receive the data must already exist.

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 must not be modified.
The following statement creates a database called `auditlogs97`:

```
CREATE DATABASE auditlogs97
```

You can also create an ANSI-compliant database. Although an ANSI-compliant database has the additional overhead of logging, its treatment of table permissions or access privileges makes it attractive in a secure environment. For more information about UNIX permissions or Windows access privileges, see "Revoking and granting privileges to protect audit data" on page 9-3.
The following SQL statement creates an ANSI-compliant database:

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

Creating a table for audit data

To load data files into a database with **dbload**, a table to receive the data must already exist.

Create a table to hold audit data with the CREATE TABLE statement. The order and data types of the columns is important.
Use the order shown in the example in the following example. The sample schema reflects the format of the **dbload** data file that **onshowaudit** created.
The sample CREATE TABLE statement in the following example creates an audit table with the name frag_logs. For information about the contents of each column,

see “Interpreting data extracted from audit records” on page 9-6.
The sample CREATE TABLE statement in the following example does not include the WITH CRCOLS option, which is for conflict resolution during database replication. To replicate the audit database, use WITH CRCOLS in the CREATE TABLE statement.

```
CREATE TABLE frag_logs
(
    adttag CHAR(4) NOT NULL,
    date_time DATETIME YEAR TO FRACTION(3) NOT NULL,
    hostname VARCHAR(128) NOT NULL,
    pid INTEGER NOT NULL,
    server VARCHAR(128) NOT NULL,
    username VARCHAR(32) NOT NULL,
    errno INTEGER NOT NULL,
    code CHAR(4) NOT NULL,
    dbname VARCHAR(128),
    tabid INTEGER,
    objname VARCHAR(128),
    extra_1 INTEGER,
    partno INTEGER,
    row_num INTEGER,
    login VARCHAR(32),
    flags INTEGER,
    extra_2 VARCHAR(160)
);
```

The table that the statement in the preceding example 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 *IBM Informix Performance Guide*.

Creating a command file for dbload

To load the audit information into the table that you created:

First create an ASCII command file for the **dbload** utility. This command file must specify the number of columns and the field delimiter that are used in the data file that **onshowaudit** created. For a description of command files and their use with **dbload**, see the *IBM Informix Migration Guide*.
Include the following information when you create the command file for **dbload**:

Delimiter
|

Number of columns
17

Table name
Table you created to receive the data

Data file name
Output file you create (to serve as input for **dbload**)

The following example uses the FILE statement to create a command file for **dbload**. The example includes the records_pat data file created in “Creating a data file for dbload” on page 9-4 and the frag_logs table created in “Creating a table for audit data” on page 9-4.

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

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

Loading audit data into a database

After you have the database, table, data, and command files for audit analysis:

Use the **dbload** command to load the audit data into the table.
The following example runs the commands specified in the `user_records` command file to load data into the **auditlogs97** database created in "Creating a database for audit data" on page 9-4:

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

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

Interpreting data extracted from audit records

When you create a database table to contain audit records that you extract from audit files, you provide a column for each field in the audit record. Table 9-2 lists recommended column names that are used in "Creating a table for audit data" on page 9-4 and describes the information that each column contains.

Table 9-2. Audit-event columns in database table for SQL access

Column Name	Description
adttag	ONLN
date_time	The date and time of the audited event
hostname	The database server name
pid	The process ID
server	The database server name
username	The username associated with the audited event
errno	The error number, if any
code	The error code, if any
dbname	The name of the database
tabid	The ID number of the affected table
objname	The index name and the table name, or similar identifier (Not in audit tables created with Informix database servers before Version 7.0)
extra_1	Information specific to the object and event, as shown in Chapter 12, "Audit event codes and fields," on page 12-1
partno	Fragmentation information (Not in audit tables created with Informix database servers before Version 7.0)
row_num	The physical row number in the affected table, which combines the row ID and the old row ID and identifies each row for the events Read Row (RDRW), Insert Row (INRW), Update Current Row (UPRW), and Delete Row (DLRW)
login	The user login name
flags	The flag set for the event, as shown in Chapter 12, "Audit event codes and fields," on page 12-1
extra_2	Information determined by the flag.

Chapter 10. The onaudit utility

Use the **onaudit** utility to manage audit masks and auditing configuration.

The **onaudit** utility manages audit masks and auditing configuration. It performs the following operations:

- Displays audit masks
- Adds audit masks
- Modifies audit masks
- Deletes audit masks
- Shows the audit configuration
- Enables and disables auditing

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

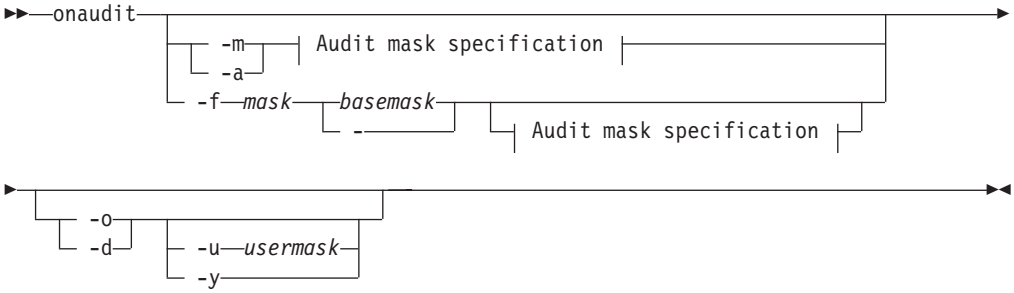
If your system has role separation enabled, only the DBSSO or AAO have the authority to run **onaudit** commands. The DBSSO has the authority to run **onaudit** functions that involve audit masks, while the AAO has the authority to run **onaudit** commands that involve audit configuration parameters. Without role separation, the user **informix** is the only user with the authority to update the `adtcfg` file or run **onaudit** commands.

Changes that the DBSSO makes to audit masks become effective immediately for user sessions.

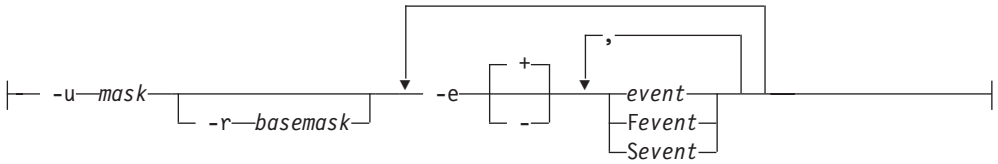
The onaudit utility: Configure audit masks

Use the **onaudit** utility to add, modify, delete and display audit masks.

Syntax



Audit mask specification:



Element	Purpose	Key Considerations
`-a`	Adds an audit mask.	None.
`-f`	Loads an input file containing a list of audit masks to be added to the audit trail.	The file must use the correct input-file format.
`-d`	Specifies that an audit mask will be deleted.	None.
`-m`	Modifies an existing audit mask.	None.
`-o`	Outputs a list of all the audit masks that have been configured in the database server.	None.
`-r` *basemask*	Specifies the name of an existing *basemask* from which you can derive events to apply to a new *targetmask*.	Subsequent changes to the *basemask* are not be reflected in the target audit masks. If no *basemask* is specified and no events are specified with the `-e` option an empty target mask is created.
`-e`	Indicates that audit events are to be added or removed from the specified *targetmask*.	Events specified as arguments to `-e` override events listed in any base mask specified with the `-r` option.
`-u` F*usermask*	Names a specific mask.	`_default`, `_require`, and `_exclude` are keywords in the system, and can be one of these names for your template or user mask. The server processes the audit mask in the predefined order. The *usermask* is limited to 32 or fewer bytes.
`-y`	Automatically responds yes to the confirmation prompt.	None.
event	Specifies an event to audit, whether the event execution succeeds or fails.	The *event* must be listed in Chapter 12, "Audit event codes and fields," on page 12-1.
F*event*	Specifies that only failed event attempts are to be audited.	The *event* must be listed in Chapter 12, "Audit event codes and fields," on page 12-1.
S*event*	Specifies that only successful event attempts are to be audited.	The *event* must be listed in Chapter 12, "Audit event codes and fields," on page 12-1.

Usage

Before you try to run the **onaudit** utility to manipulate audit masks, ensure that the server is running, and that you hold the DBSSO role.

All the options of this utility must be entered as shown because they are case-sensitive.

For a high-availability cluster, the audit mask must be created on the primary server. All of the servers in the cluster use the audit mask on the primary server.

Run the **onaudit** command with the `-a` option when you want to add one or more audit masks to an audit trail. Note that _default, _require, and _exclude are keywords that the server understands and processes in a particular order.

Attention: Even though_default, _require, and _exclude are stored as keywords in the system they are not automatically defined. You must explicitly create and add events to them before trying to use these audit masks.

Run the **onaudit** command with the `-f` option to load an existing input file that contains a listing of audit masks. The format of this input file's contents is:

<mask_name> <base_mask> <event_list>

A hyphen (-) is used in places where the base mask is unavailable.

Run the `-d` option of the **onaudit** command to delete a specified audit mask. When you select the `-d` option of the **onaudit** utility:

- The `-y` option is used to respond yes to all prompts.
- If the `-u` *mask* option is omitted, all masks are deleted, including the _default, _require, and _exclude masks.
- If the `-y` or the `-u` options are omitted, the **onaudit** utility requests confirmation that this is intentional so that you do not accidentally delete all user masks.

Use the `-m` option of the **onaudit** command when you must modify an existing audit mask. Use a plus (+) sign to add an event to an audit mask or use the hyphen (-) sign to delete an event from a mask. Use a comma (,) to separate multiple events that are being added to the mask. Do not add any spaces between the comma and the event mnemonics.

If no sign is specified before an event mnemonic, the event is added to the mask.

The `-o` option of the **onaudit** command sends information about the mask to standard output. When you select the `-o` option of the **onaudit** utility:

- The `-y` option is used to respond yes to all prompts.
- If the `-u` *mask* option is omitted, all masks are displayed.
- If the `-y` or the `-u` options are omitted, **onaudit** requests confirmation before it displays all the masks because it can result in the display of large amounts of data.

The output file is displayed in the following format, which is identical to the format of input files:

<mask_name> <base_mask> <event_list>

A hyphen (-) is used in places where the base mask is unavailable.

Run the command with the `-r` option to copy all of the events associated with the specified base mask (which can be a system mask) to a new target mask.

The -u option of the **onaudit** command can be used in combination with the -a, -d, -m, and -o options.

Examples

Example 1: Add an audit mask

The following example creates a template mask named pat with events CRTB (CREATE TABLE) and RVLB (REVOKE SECURITY LABEL) defined. The -a option is used to create the mask. The -u option is used to identify the mask name. The -e option is used to list the events defined in the mask.

```
onaudit -a -u pat -e +CRTB,RVLB
```

Example 2: Load a file containing one or more audit masks

The following example loads the masks defined in the input file entitled, masks_feb.

```
onaudit -f /work/masks_feb
```

Example 3: Delete an audit mask

The following example shows how to delete the _default audit mask:

```
onaudit -d -u _default
```

Example 4: Modify an audit mask

The following example modifies the _default audit mask by adding the GRXM (GRANT EXEMPTION) event and deleting the CRTB (CREATE TABLE) event:

```
onaudit -m -u _default -e +GXRM, -e -CRTB
```

Example 5: Display an audit mask

The following example shows how to display the audit mask for the user pat, indicating that the individual user mask contains the audit events LKTB (LOCK TABLE), CRTB (CREATE TABLE), and failed attempts to ADCK (ADD CHUNK):

```
onaudit -o -u pat
```

The following example is the output of the sample command:

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

Example 6: Derive an audit mask

The following example creates a new user mask named pat. The new mask derives the events specified in the _secureL template mask, but excludes RDRW (READ ROW) and includes LKTB (LOCK TABLE), successful attempts to ADCK (ADD CHUNK), and all attempts to CRTB (CREATE TABLE):

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

Related concepts:

"Audit masks" on page 7-1

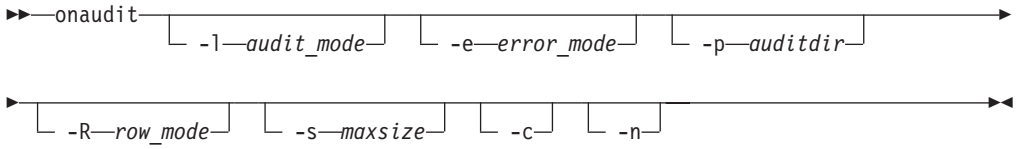
Related tasks:

"Adding one or more masks using an input file" on page 8-11

The onaudit utility: Configure auditing

Use the **onaudit** utility to start, stop, and configure auditing.

Syntax



Element	Purpose	Key Considerations
-c	Shows the current audit configuration as the values of the auditing configuration parameter in the ADTCFG file.	None.
-e *error_mode*	Specifies the error-handling method for auditing when a record cannot be written to the audit file or event log: • 0 = Continue processing the thread and record the error in the message log. Errors for subsequent attempts to write to the audit file are also sent to the message log. • 1 = Suspend processing a thread when the database server cannot write a record to the current audit file. The database server attempts to write the record until it succeeds. • 3 = Shut down the server.	This option sets the **ADTERR** configuration parameter in the ADTCFG file. You can use this option only when auditing is enabled.
-l *audit_mode*	Specifies the audit mode: • 0 = Disable auditing • 1 = Audit all sessions • 3 = Audit DBSSO actions • 5 = Audit database server administrator actions • 7 = Audit DBSSO and database server administrator actions	This option sets the **ADTMODE** configuration parameter in the ADTCFG file.
-n	Starts a new audit file.	You can use this option only when auditing is enabled.
-p *auditdir*	Specifies a new directory in which the database server creates audit files. The change occurs with the next write attempt. The database server creates a new audit file in the new directory, beginning with the first available number that is equal to or greater than 0.	This option sets the **ADTPATH** configuration parameter in the ADTCFG file. You can use this option only when auditing is enabled.

Element	Purpose	Key Considerations
-R *row_ mode*	Controls selective row-level auditing: • 0 = Selective row-level auditing is disabled. • 1 = Selective row-level auditing is enabled for tables that are set with the AUDIT flag. • 2 = Selective row-level auditing is enabled for tables that are set with the AUDIT flag. The primary key, if it is an integer data type, is included in the audit records.	This option sets the **ADTROWS** configuration parameter in the ADTCFG file.
-s *maxsize*	Specifies the maximum size (in bytes) of an audit file. 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, the size is set automatically to the minimum value. When an audit file reaches or exceeds the maximum size, the database server closes the current file and starts a new audit file.	This option sets the **ADTSIZE** configuration parameter in the ADTCFG file. You can use this option only when auditing is enabled.

Usage

Before you try to run the **onaudit** utility, ensure that the server is running, that an audit mask with defined audit events has been added, and that you hold the AAO role.

All the options of this utility must be entered as shown because they are case-sensitive.

The **onaudit** command takes effect immediately for all new and existing user sessions.

To enable auditing for a high-availability cluster, you must enable auditing on the primary server and on every secondary server in the cluster. The audit mask must be created on the primary server. All of the servers in the cluster use the audit mask set on the primary server. Audit records for insert, update, and delete operations are created only on the primary server.

You can start auditing by using the **onaudit** command with the -l option set to a positive value. You can specify whether to limit auditing to certain tables by using the -R option. A new audit file is created when you enable auditing. When you start auditing with the **onaudit** command, the audit file size, the error mode, and the audit file directory information in the ADTCFG file is used.

You can stop auditing by using the onaudit -l 0 command. The database server stops auditing all existing sessions, and does not audit new sessions.

You can view the current audit configuration by using the onaudit -c command. That command displays the contents of the ADTCFG file.

You can dynamically change the behavior of auditing by using the **onaudit** command with any of its options.

You can use the -n option to create a new audit file:

- For database server-managed auditing, the **onaudit** utility closes the current database server audit file, stores it in the specified directory, and creates 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 `maple.123`, the next audit file is `maple.124`.
- For operating-system-managed files, the **onaudit** utility closes the current operating-system audit file, stores it as part of the operating-system audit trail, and creates a new audit file. For the naming conventions for files in the audit trail, see your operating-system documentation.

Examples

Example 1: Start auditing

The following command starts auditing all sessions:

```
onaudit -l 1
```

Example 2: Stop auditing

The following command stops auditing all current sessions. Also, sessions started after the command is run are not audited:

```
onaudit -l 0
```

Example 3: Change the audit configuration

The following command changes the error mode to 3 (shut down the server), the auditing mode to 3 (Audit DBSSO actions), and starts a new audit file:

```
onaudit -e 3 -l 3 -n
```

Example 4: Audit selected tables

The following command continues auditing all tables that have the AUDIT flag and stops auditing all other tables:

```
onaudit -R 1
```

Related concepts:

"Specifying a directory for the audit trail (UNIX)" on page 8-6
"Setting the error mode" on page 8-6
"Setting the audit level" on page 8-7
"Activating auditing" on page 8-9

Related tasks:

"Setting up selective row-level auditing" on page 8-8
"Displaying the audit configuration" on page 8-13
"Starting a new audit file" on page 8-14
"Changing audit levels" on page 8-15
"Changing the audit error mode" on page 8-15
"Turning off auditing" on page 8-15

Related reference:

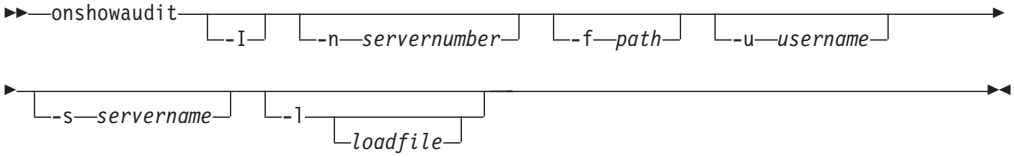
Chapter 13, "The ADTCFG file," on page 13-1

Chapter 11. The onshowaudit utility

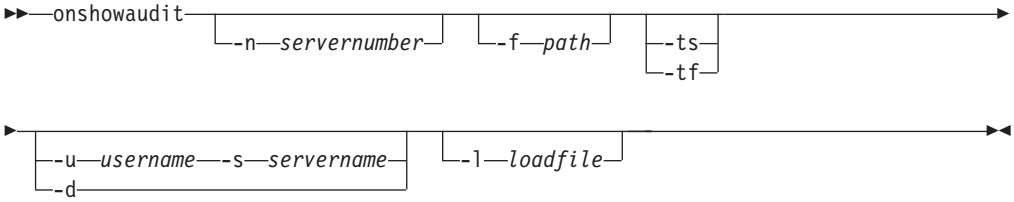
Use the **onshowaudit** utility to view the audit information from an existing audit trail. You can use this command to extract information for a particular user, database server, or both, making it possible to isolate a particular subset of data from a potentially large audit trail.

Syntax

UNIX:



Windows:



Element	Purpose	Key Considerations
`-d`	Indicates that the **onshowaudit** utility must use default values for the user (**current user**) and database server (**INFORMIXSERVER**) fields.	This option is only available on the Windows operating system.
`-f` *path*	Specifies an audit trail to examine, only for database server-managed auditing.	The path can be a full path or just a file name. If this option is omitted, or if *path* is only a file name, see the notes that immediately follow this table.
`-I`	Indicates that the specified audit trail is for the database server. **Note:** This option is a holdover from a time when operating system (OS) auditing was supported. The **-I** must be included for compatibility.	This option is case-sensitive. The UNIX operating system uses the Informix database server audit trail

Element	Purpose	Key Considerations
-l	Directs **onshowaudit** to extract information with delimiters so that it can be redirected to a file or pipe and loaded into a database table or other application that accepts delimited data.	When using the Windows operating system you must remove the six header lines that are in the output file before you use that file as input for **dbload** or for an external file. On the Windows operating system, you must enter a load file name argument for the -l option. On theUNIX operating system this file name argument is optional. On the UNIX operating system, if you do not specify a file name, the output is routed to standard output.
-n *servernumber*	Extracts audit records from the **ADTPATH** location specified in the adtcfg.*servernumber* file.	If the adtcfg.*servernumber* file does not exist, the contents of the ADTCFG file are used for audit configuration.
-tf	Displays only failure audit records	This option is only available on the Windows operating system.
-ts	Displays only success audit records	This option is only available on the Windows operating system.
-s *servername*	Specifies which database server must have audit information extracted.	None.
-u *username*	Specifies the login name of a user for extraction of audit information.	None.

Usage

The **onshowaudit** utility performs the following operations:

- Extracts audit information from an audit trail
- Prepares extracted audit data for the **dbload** utility

The **onshowaudit** command extracts data from an audit trail but does not process the records or delete them from the audit trail. You must only access the audit trail with the **onshowaudit** command because it includes certain protections.

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

By default, the **onshowaudit** command is displayed to the standard output (your screen). You can redirect the formatted output to a file or pipe and can specify that the **onshowaudit** command reformat the output so that you can load it into an Informix database table.

If you modify the audit configuration with the **onaudit** utility, the adtcfg.*servernumber* file stores the changed configuration. If the server audit configuration is modified, use the -n option to specify the server number for **onshowaudit**. Using the -n option allows **onshowaudit** to read the right **ADTPATH**

stored in adtcfg.*servernumber* file. The **onshowaudit** utility extracts data from all the audit files it finds that are in sequence, starting with the lowest integer.

If only a file name is specified, the utility searches the ADTPATH directory for that file and extracts audit data from it.

If a complete path name is specified, the utility extracts audit data from the named file.

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

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

If -f is omitted, **onshowaudit** searches for audit files in the ADTPATH directory specified in the default ADTCFG file. The -f *path* option specifies the directory and file name of the audit files. The audit directory and file name must conform to minimum security levels. The directory must be owned by user **informix**, belong to the AAO group, and must not allow public access (0770 permission). The files must have comparable permissions (0660 permission). The files must not be symbolic links to other locations. The directory can be a symbolic link. If the audit directory and files are not secure, the **onshowaudit** utility returns an error message and does not display the audit results.

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

Examples

Example 1: Reading a specific audit log file

The following command shows the audit log file /work/aaodir/ol_lx_rama.7:

```
onshowaudit -I -f /work/aaodir/ol_lx_rama.7
```

Example 2: Filtering audit records by user

The following command shows only the records that pertain to usr1 in the audit log file /work/aaodir/ol_lx_rama.7:

```
onshowaudit -I -f /work/aaodir/ol_lx_rama.7 -u usr1
```

Example 3: Filtering audit records by server name

The following command shows only the records that pertain to usr1 on the ol_lx_rama server in the audit log file /work/aaodir/ol_lx_rama.7:

```
onshowaudit -I -f /work/aaodir/ol_lx_rama.7 -u usr1 -s ol_lx_rama
```

Related reference:

Chapter 13, “The ADTCFG file,” on page 13-1

Chapter 12. Audit event codes and fields

The secure-auditing facility audits certain database server events.

If you are using the **onshowaudit** utility, auditable events on each database server generate event codes. These codes represent actions on the server that can indicate possibly illegitimate usage or tampering.

Important: The Informix secure-auditing facility audits only the events that the following table lists. You might encounter additional SQL statements that the secure-auditing facility does not audit.

Table 12-1 shows the audit-event information in alphabetic order by event code:

- The Event Code column has the acronym that database server utilities use to identify audit events.
- The Event column shows the event name.
- The Variable Contents column has other categories of **onshowaudit** information that are displayed for the event on that row. The categories of information are:
  - tabid
  - dbname
  - objname
  - extra_1
  - partno
  - row_num
  - login
  - flags
  - extra_2

  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.
- The Notes section after the table provides more information about some of the entries in the **Variable Contents** column.

Tip: Granted lists can be long for 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.

Table 12-1. Audit events listed by event code

Event Code	Event	Variable Contents
ACTB	Access Table	dbname: *database_name* tabid: *owner_name*, *table_id*
ADCK	Add Chunk	dbname: *dbspace*, *name* extra_1: *offset* flags: *mirror_status*[1] extra_2: *path* and *size*

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
ADLG	Add Transaction Log	dbname: *dbspace*, *name* extra_1: *log_size*
ALFR	Alter Fragement	dbname: *database_name* tabid: *table_id* objname: *index_name* extra_1: *operation_type*[18] login: *owner* flags: *frag_flags*[15] extra_2: *dbspaces* *alter_type*: 0 = normal, 1 = forced alter
ALIX	Alter Index	dbname: *database_name* tabid: *table_id* login: *owner*[14] flags: *cluster_flag*[9,14] extra_2: *index_name*[14]
ALLC	Alter Security Label Component	dbname: *database_name* objname: *component_name* extra_2: *component_type*
ALME	Alter Access Method	dbname: *database_name* tabid: *access*, *method_ID* objname: *access_method*, *name* login: *access_method*, *owner*
ALOC	Alter Operator Class	dbname: *database_name* extra_1: *cluster_size* login: *owner* extra_2: *cluster_name*
ALOP	Alter Optical Cluster	dbname: *database_name* extra_1: *cluster_size* login: *owner* extra_2: *cluster_name*
ALSQ	Alter Sequence	dbname: *database_name* tabid: *table_id*

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
ALTB	Alter Table	dbname: *database_name* tabid: *old_table_id* extra_1: *new_table_id*[14] partno: *frag_id* extra_2: *new_part_number_list*[14]
ALTX	Alter trusted context	dbname: *database_name* objname: *context_name* login: *system_authid*
ALUR	Alter User	objname: *user_name*
BGTX	Begin Transaction	none
CLDB	Close Database	dbname: *database_name*
CMTX	Commit Transaction	none
CRAG	Create Aggregate	dbname: *database_name* objname: *aggregate_name* login: *owner*
CRAM	Create Audit Mask	login: *user_id*
CRBS	Create Storage Space	dbname: *storage_name*, *space_name* login: *owner* flags: *mirror_status*[1] extra_2: *media*
CRBT	Create Opaque Type	dbname: *database_name* objname: *opaque_type_name* login: *opaque_type*, *owner*
CRCT	Create Cast	dbname: *database_name* tabid: *from_type_ID* objname: *function_name* or "-" extra_1: *from_type_xid* partno: *to_type_ID* row_num: *to_type_xid* login: *function_owner* or "-"
CRDB	Create Database	dbname: *dbspace* extra_2: *database_name*
CRDS	Create Dbspace	dbname: dbspace, *name* flags: *mirror_status*[1]

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
CRDT	Create Distinct Type	dbname: *database_name* objname: *distinct_type_name* login: *distinct_type*, *owner*
CRIX	Create Index	dbname: *database_name* tabid: *table_id* objname: *index_name* login: *owner* flags: *frag_flags*[15] extra_2: *dbspace_list*
CRLB	Create Security Label	dbname: *database_name* objname: *policy.label_name*
CRLC	Create Security Label Component	dbname: *database_name* objname: *component_name*
CRME	Create Access Method	dbname: *database_name* tabid: *access_method_ID* objname: *access_method_name* login: *access_method_owner*
CROC	Create Operator Class	dbname: *database_name* tabid: *operator_class_ID* objname: *operator_class_name* login: *owner*
CROP	Create Optical Cluster	dbname: *database_name* tabid: *table_id* extra_1: *cluster_size* login: *owner* extra_2: *cluster_name*
CRPL	Create Security Policy	dbname: *database_name* objname: *policy_name*
CRPT	Decryption Failure or Attempt	dbname: *database_name* objname: *statement*
CRRL	Create Role	dbname: *database_name* objname: *rolename*

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
CRRT	Create Named Row Type	dbname: *database_name* tabid: *row_type_xid* objname: *named_row_type_name* login: *named_row_type_owner*
CRSN	Create Synonym	dbname: *database_name* tabid: *synonym_table_id* extra_1: *base_table_id* login: *owner* flags: *synonym_type*[7] extra_2: *synonym_name*
CRSP	Create SPL Routine	dbname: *database_name* tabid: *proc_id* login: *owner* extra_2: *procedure_name*
CRSQ	Create Sequence	dbname: *database_name* tabid: *table_id* objname: *owner*
CRTB	Create Table	dbname: *database_name* tabid: *table_id* objname: *owner* login: *table_name* flags: *frag_flags*[15] extra_2: *dbspace_list*
CRTR	Create Trigger	dbname: *database_name* tabid: *table_id* row_num: *trigger_id*[14] login: *owner*[14] extra_2: *trigger_name*[14]
CRTX	Create trusted context	dbname: *database_name* objname: *context_name* login: *system_authorization_id*
CRUR	Create User	objname: *user_name*

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
CRVW	Create View	dbname: *database_name* tabid: *view_table_id* login: *owner* extra_2: *view_name*
CRXD	Create XADatasource	dbname: *database_name* objname: *owner* objname: *XA_data_source_name*
CRXT	Create XADatasource Type	dbname: *database_name* objname: owner objname: *XA_data_source_type_name*
DLRW	Delete Row	dbname: *database_name* tabid: *table_id* extra_1: *part_number* partno: *frag_id* row_num: *row_number*[14]
DNCK	Bring Chunk Offline	extra_1: *chunk_number* flags: *mirror_status*[1]
DNDM	Disable Disk Mirroring	extra_1: *dbspace_number*
DRAG	Drop Aggregate	dbname: *database_name* objname: *aggregate_name* login: *owner*
DRAM	Delete Audit Mask	login: *user_id*
DRBS	Drop Storage Space	dbname: *storage_space_name*
DRCK	Drop Chunk	dbname: *dbspace_name* flags: *mirror_status*[1] extra_2: *path*
DRCT	Drop Cast	dbname: *database_name* tabid: *from_type_ID* extra_1: *from_type_xid* partno: *to_type_ID* row_num: *to_type_xid*
DRDB	Drop Database	dbname: *database_name*
DRDS	Drop Dbspace	dbname: *dbspace_name*

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
DRIX	Drop Index	dbname: *database_name* tabid: *table_id* login: *owner* extra_2: *index_name*
DRLB	Drop Security Label	dbname: *database_name* objname: *policy.label_name*
DRLC	Drop Security Label Component	dbname: *database_name* objname: *component_name*
DRLG	Drop Transaction Log	extra_1: *log_number*
DRME	Drop Access Method	dbname: *database_name* tabid: *access_method_ID* objname: *access_method_name* login: *access_method_owner*
DROC	Drop Operator Class	dbname: *database_name* objname: *operator_class_name* login: *owner*
DROP	Drop Optical Cluster	dbname: *database_name* login: *owner* extra_2: *cluster_name*
DRPL	Drop Security Policy	dbname: *database_name* objname: *policy_name*
DRRL	Drop Role	dbname: *database_name* objname: *role_name*
DRRT	Drop Named Row Type	dbname: *database_name* tabid: *dropped_type_xid*
DRSN	Drop Synonym	dbname: *database_name* tabid: *synonym_table_id* login: *owner* extra_2: *synonym_name*
DRSP	Drop SPL Routine	dbname: *database_name* login: *owner* extra_2: *spname*
DRSQ	Drop Sequence	dbname: *database_name* tabid: *table_id*

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
DRTB	Drop Table	dbname: *database_name* tabid: *table_id* objname: *table_name* login: *owner* flags: *drop_flags*[21] extra_2: *part_number_list*
DRTR	Drop Trigger	dbname: *database_name* row_num: *trigger_id* login: *owner* extra_2: *trigger_name*
DRUR	Drop User	objname: *user_name*
DRTX	Drop trusted context	objname: *context_name*
DRTY	Drop Type	dbname: *database_name* objname: *type_name* login: *type_owner*
DRVW	Drop View	dbname: *database_name* tabid: *view_table_id* flags: *drop_flags*[21]
DRXD	Drop XADatasource	dbname: *database_name* objname: *owner* objname: *XA_data_source_name*
DRXT	Drop XADatasource Type	dbname: *database_name* objname: *owner* objname: *XA_data_source_type_name*
EXSP	Execute SPL Routine	dbname: *database_name* tabid: *proc_id*
GRDB	Grant Database Access	dbname: *database_name* extra_1: *privilege*[5] extra_2: *grantees*[4]
GRDR	Grant Default Role	dbname: *database_name* objname: *role_name* login:*grantor* extra_2: *grantees*[4]

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
GRFR	Grant Fragment Access	dbname: *database_name* tabid: *table_id* objname: *fragment* extra_1: *privilege*[5, 14] login: *grantor* extra_2: *grantees*[4, 14]
GRLB	Grant Security Label	dbname: *database_name* objname: *policy.label_name* login: *grantee*[4] extra_2: *access_type*
GRRL	Grant Role	dbname: *database_name* objname: *role_name* login: *grantor* extra_2: *grantees*[4]
GRSA	Grant DBSECADM	login: *grantee*
GRSS	Grant SETSESSIONAUTH	dbname: *database_name* login: *grantee* extra_2: *surrogate_user_list*
GRTB	Grant Table Access	dbname: *database_name* tabid: *table_id* extra_1: *privilege*[5, 14] login: *grantor* extra_2: *grantee*[4, 14], *update_columns*, *select_columns*[4, 14]
GRXM	Grant Exemption	dbname: *database_name* objname: *policy_name* login: *grantee* extra_2: *rule*
INRW	Insert Row	dbname: *database_name* tabid: *table_id* partno: *frag_id* row_num: *row_id*
LGDB	Change Database Log Mode	dbname: *database_name* flags: *log_status*[6]

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
LKTB	Lock Table	dbname: *database_name* tabid: *table_id* flags: *lock_mode*[8]
LSAM	List Audit Masks	none
LSDB	List Databases	none
MDLG	Modify Transaction Logging	flags: *buffered_log_flags*[2]
ONAU	**onaudit**	extra_2: *command_line*
ONBR	**onbar**	extra_2: *command_line*
ONCH	**oncheck**	extra_2: *command_line*
ONIN	**oninit**	extra_2: *command_line*
ONLG	**onlog**	extra_2: *command_line*
ONLO	**onload**	extra_2: *command_line*
ONMN	**onmonitor**	extra_2: *command_line*
ONMO	**onmode**	extra_2: *command_line*
ONPA	**onparams**	extra_2: *command_line*
ONPL	**onpload**	extra_2: *command_line*
ONSP	**onspaces**	extra_2: *command_line*
ONST	**onstat**	extra_2: *command_line*
ONTP	**ontape**	extra_2: *command_line*
ONUL	**onunload**	extra_2: *command_line*
OPDB	Open Database	dbname: *database_name* flags: *exclusive_flag* extra_2: *database_password*
OPST	Optimize Storage	fragment <parameters>: *part_numbers* table <parameters>: *table_name*:*database_name*:*owner_name* compression purge_dictionary: *date*
PWUR	Set User Password	objname: *user_name*
RBSV	Rollback to Savepoint	dbname: *database_name* extra_1: *transaction_id* objname: *savepoint_name*
RDRW	Read Row	dbname: *database_name* tabid: *table_id* extra_1: *part_number* partno: *frag_id* row_num: *row_id*[14]

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
RLOP	Release Optical Cluster	dbname: *family_name* row_num: *volume_number*
RLSV	Release Savepoint	dbname: *database_name* extra_1: *transaction_id* objname: *savepoint_name*
RLTX	Rollback Transaction	none
RMCK	Clear Mirrored Chunks	extra_1: *dbspace_number*
RNUR	Rename User	objname: *old_user_name* extra_2: *new_user_name*
RNDB	Rename Database	dbname: *database_name* objname: *new_dbname* login: *user_id*
RNDS	Rename dbspace	dbname: *dbspace_name* objname: *new_dbspace_name*
RNIX	Rename Index	dbname: *index_name* objname: *new_index_name*
RNLB	Rename Security Label	dbname: *database_name* objname: *old_policy.label_name* extra_2: *new_policy.label_name*
RNLC	Rename Security Label Component	dbname: *database_name* objname: *old_component_name* extra_2: *new_component_name*
RNPL	Rename Security Policy	dbname: *database_name* objname: *old_policy_name* extra_2: *new_policy_name*
RNSQ	Rename Sequence	dbname: *database_name* tabid: *table_id*
RNTC	Rename Table/Column	dbname: *database_name* tabid: *table_id* objname: *new_table/column_name* extra_1: *colno(*)* login: *owner* extra_2: *table_name(**)*

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
RNTX	Rename trusted context	objname: *context_name* extra_2: *new_context name*
RSOP	Reserve Optical Cluster	dbname: *family_name* row_num: *volume_number*
RSOP	Reserve Optical Cluster	dbname: *family_name* row_num: *volume_number*
RVDR	Revoke Default Role	dbname: *database_name* objname: *role_name* login: *revoker* extra_2: *revokees*[4]
RVFR	Revoke Fragment Access	dbname: *database_name* tabid: *table_id* objname: *fragment* extra_1: *privilege*[5, 14] login: *revoker* extra_2: *revokees*[4, 14]
RVLB	Revoke Security Label	dbname: *database_name* objname: *policy.label_name* login: *grantee* extra_2: *access_type*
RVRL	Revoke Role	dbname: *database_name* objname: *role_name* login: *revoker* extra_2: *revokees*[4]
RVSA	Revoke DBSECADM	login: *grantee*
RVSS	Revoke SETSESSIONAUTH	dbname: *database_name* login: *grantee* extra_2: *surrogate_user_list*
RVTB	Revoke Table Access	dbname: *database_name* tabid: *table_id* extra_1: *privilege*[5, 14] login: *revoker* flags: *drop_flags*[21] extra_2: *revokees*[4, 14]

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
RVXM	Revoke Exemption	dbname: *database_name* objname: *policy_name* login: *grantee* extra_2: *rule*
SCSP	System Command, SPL Routine	extra_2: *command_string*
STCO	Set Collation	dbname: *database_name* objname: *locale_name*
STCN	Set Constraint	dbname: *database_name* flags: *constraint_mode*[11] extra_2: *constraint_names*
STDF	Set Debug File	dbname: *database_name* extra_2: *file_path*
STDP	Set Database Password	dbname: *database_name* login: *user_id*
STDS	Set Dataskip	flags: skip flags[16] extra_2: *dbspace_list*
STEP	Set Encryption Password	dbname: *database_name*
STEV	Set Environment	objname: *environment_variable_and_value*
STEX	Set Explain	flags: *explain_flags*[12]
STIL	Set Isolation Level	extra_1: *isolation_level*[3]
STLM	Set Lock Mode	flags: *wait_flags*[13]
STNC	Set No Collation	dbname: *database_name* objname: *locale_name*
STOM	Set Object Mode	dbname: *database_name* tabid: *table_id* extra_1: *command_mode_flag*[23] flags: *object_type_flag*[2 4] extra_2: *object_names*
STOP	Stop Violations	dbname: *database_name* tabid: *table_id*
STPR	Set Pdqpriority	flags: *priority_level*[17]
STRL	Set Role	dbname: *database_name* objname: *role_name*

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
STRS	Set Resident	dbname: *database_name* objname: *fragment_list* extra_1: *fragment_information*
STRT	Start Violations	dbname: *database_name* tabid: *table_id* extra_1: *Vio_tid* flags: *Dia_tid*
STSA	Set Session Authorization	dbname: *database_name* login: *new_user_name*
STSC	Set Statement Cache	objname: *statement_name*
STSN	Start New Session	none
STSV	Set Savepoint	dbname: *database_name* extra_1: *transaction_id* objname: *savepoint_name*
STTX	Set Transaction Mode	extra_1: *operation*[20] flags: *mode_flags*[19] extra_2:
SVXD	Save External Directives	dbname: *database_name* objname: *active/inactive/test* objname: *directive_text*
TCTB	Truncate Table	dbname: *database_name* tabid: *table_id* objname: *table_name*
TMOP	Time Optical Cluster	flags: time flag[13]
ULTB	Unlock Table	dbname: *database_name* tabid: *table_id*
UPAM	Update Audit Mask	login: user id
UPCK	Bring Chunk Online	extra_1: *chunk_number* flags: *mirror_status*[1]
UPDM	Enable Disk Mirroring	extra_1: *dbspace_number*

Table 12-1. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
UPRW	Update Current Row	dbname: *database_name* tabid: *table_id* extra_1: *old_part_number* row_num: *old_row_id*[14] flags: *new_row_id* extra_2: *new_part_number*
USSP	Update Statistics, SPL Routine	dbname: *database_name* tabid: *proc_id*
USTB	Update Statistics, Table	dbname: *database_name* tabid: *table_id*

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 bytes, the audit processing described in "Audit analysis with SQL" on page 9-3 truncates the lists to 166 bytes.
5. Database Privileges:

   Table-Level Privileges:
   1 Select
   2 Insert
   4 Delete
   8 Update
   16 Alter
   32 Index
   64 Reference
   4096 Execute Procedure (When Grant privilege is executed. tabid is 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 (frag) 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

Chapter 13. The ADTCFG file

This chapter contains a list of the configuration parameters in the ADTCFG file and a short explanation of each configuration parameter.

Note: When any changes are made to the audit configuration, the server stores the changed configuration settings to the adtcfg.*servernumber* file. The server then reads the parameters in the adtcfg.*servernumber* file instead of the ADTCFG file.

Each configuration parameter has one or more of the following attributes (depending on their relevance):

default value
Default value that is in the adtcfg.std file

if not present
Value that is supplied if the parameter is missing from your ADTCFG file

units Units in which the parameter is expressed

separators
Separators that can be used when the parameter value has several parts. Do not use white space within a parameter value

range of values
Valid values for this parameter

takes effect
Time at which a change to the value of the parameter actually affects the operation of the database server

utility Name of the command-line utility that you can use to change the value of the parameter

Related reference:

"The onaudit utility: Configure auditing" on page 10-5

Chapter 11, "The onshowaudit utility," on page 11-1

ADTCFG file conventions

The UNIX file $INFORMIXDIR/aaodir/adtcfg or the Windows file %INFORMIXDIR%\aaodir\adtcfg is called the ADTCFG configuration file or 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 Informix configuration parameters, see the *IBM Informix Administrator's Reference*.

ADTERR configuration parameter

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

default value
0

range of values
0, 1, 3

0 = continue error mode

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

1 = halt error mode: suspend thread processing

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

3 = halt error mode: shut down system

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

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

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

ADTMODE configuration parameter

ADTMODE controls the level of auditing.

default value
0

range of values
0, 1, 3, 5, 7

0 = auditing disabled

1 = auditing on; starts auditing for all sessions

3 = auditing on; audits DBSSO actions

5 = auditing on; audits database server administrator actions

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

takes effect
When **onaudit** is run to change the value or after the server is started

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

ADTPATH configuration parameter

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

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

The **ADTPATH** values are:

default value
`/usr/informix/aaodir` (on UNIX), `%informixdir%\aaodir` (on Windows)

range of values
Any valid directory path

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

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

ADTROWS configuration parameter

Use the **ADTROWS** configuration parameter to control selective row-level auditing of tables.

default value
0

range of values
0, 1, 2

takes effect
When onaudit is run to change the value or after the database server is restarted.

utility **onaudit** (**onaudit -R** *row mode*)

Where *row mode* is set to:

- `0` for auditing row-level events on all tables
- `1` to allow control of which tables are audited. Row-level events DLRW, INRW, RDRW, and UPRW are audited only on tables for which the AUDIT flag is set.
- `2` to turn on selective row-level auditing and also to include the primary key in audit records (the primary key is only recorded if it is an integer)

see CREATE TABLE and ALTER TABLE in the *IBM Informix Guide to SQL: Syntax*

ADTSIZE configuration parameter

Use the **ADTSIZE** configuration parameter to specify 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.

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*)

Part 3. Appendixes

Appendix. Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

Accessibility features for IBM Informix products

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in IBM Informix products. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers.
- The attachment of alternative input and output devices.

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

Related accessibility information

IBM is committed to making our documentation accessible to persons with disabilities. Our publications are available in HTML format so that they can be accessed with assistive technology such as screen reader software.

IBM and accessibility

See the *IBM Accessibility Center* at http://www.ibm.com/able for more information about the IBM commitment to accessibility.

Dotted decimal syntax diagrams

The syntax diagrams in our publications are available in dotted decimal format, which is an accessible format that is available only if you are using a screen reader.

In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), the elements can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read punctuation. All syntax elements that have the same dotted decimal number (for example, all syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, the word or symbol is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is read as 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol that provides information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, that element is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 refers to a separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

? Specifies an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element (for example, 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! Specifies a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

***** Specifies a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be

repeated. For example, if you hear the line `5.1* data-area`, you know that you can include more than one data area or you can include none. If you hear the lines `3*, 3 HOST`, and `3 STATE`, you know that you can include `HOST`, `STATE`, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The `*` symbol is equivalent to a loop-back line in a railroad syntax diagram.

`+` Specifies a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times. For example, if you hear the line `6.1+ data-area`, you must include at least one data area. If you hear the lines `2+`, `2 HOST`, and `2 STATE`, you know that you must include `HOST`, `STATE`, or both. As for the * symbol, you can repeat a particular item if it is the only item with that dotted decimal number. The `+` symbol, like the `*` symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A.

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

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

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

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

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

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

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

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

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

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

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

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

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

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

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

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

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy,

modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_. All rights reserved.

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

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Index

Special characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

Printed in USA

SC27-3562-04

Spine information:

Informix Product Family Informix **Version 11.70**

IBM Informix Security Guide