

HCL Informix 14.10 - Security Guide



Contents

- Chapter 1. Security..... 3**
 - Security in HCL Informix®.....3
 - Securing data.....3
 - Auditing data security..... 159
- Index..... 259**

Chapter 1. Security

You can secure your Informix® database server and the data that is stored in your Informix® databases. You can encrypt data, secure connections, control user privileges and access, and audit data security.

Main resource

Security in HCL Informix®

These topics document methods for keeping your data secure by preventing unauthorized viewing and altering of data or database objects, including how to use the secure-auditing facility of the database server.

This information is for the following users:

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

These topics are taken from the *HCL® Informix® Security Guide*.

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.

HCL Informix® directory security

HCL 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 \(UNIX\) on page 4](#).
- 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 \(UNIX\) on page 5](#).
- 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 \(UNIX\) on page 7](#).
- You cannot continue to use many programs with the database server if a security problem in \$INFORMIXDIR or its subdirectories arises. See [Securing \\$INFORMIXDIR and its subdirectories on page 12](#)
- Most HCL® Informix® utilities run as secure users and belong to a secure group. See [Users and group membership for running utilities on page 14](#)
- The chunk files that hold the data for Informix® must be secure also. See [Security of the chunk files on page 15](#).
- 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 16](#).

Utilities for checking directory security (UNIX™)

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 \(UNIX\) on page 5](#)), 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 *HCL® 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 *HCL® Informix® Installation Guide* *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™)

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™)

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 the 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™)

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

Table 1. Secure permissions for the installation directory and subdirectories

Subdirectory	Owner	Group	Permissions (octal)
. (\$INFORMIXDIR)	informix	informix	755
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 185](#) for more information about Database Server Administrator (DBSA), Audit Analysis Officer (AAO), and Database System Security Officer (DBSSO) groups.

The onsecurity utility (UNIX™)

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 the 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.



Important: The onsecurity utility itself cannot change file permissions. It supports an extensive set of options by which you can specify how you would like the problem fixed, and upon request, it generates a script that user **root** can run to modify permissions or settings. Changes to file or directory permissions can result from user **root**



running the script that onsecurity generates, but changes to permissions or settings cannot be made directly by any onsecurity command.

Syntax

```
onsecurity { <options> path | -h | -v | -version }
```

Options

```
" [-ggroup] "
```

```
" [-uuser] "
```

```
" [-i] "
```

```
" [-n] "
```

```
" [-e] "
```

```
" [-p] "
```

```
" [-d] [{-t | -v | -q}][-r <fix actions>] "
```

Fix actions

```
" [-G { chmod | chgrp [=group] | add }] "
```

```
" [-U { chown [=user] | add }] "
```

```
" [-O { chmod | add }] "
```

Parameters


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

Element	Purpose	Key Considerations
<i>path</i>	Specifies the directory or file path that the utility analyzes.	
<i>group</i>	Specifies a group name or a group number.	
<i>user</i>	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 .	
-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 <i>group</i>	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 <code>/etc/informix</code> subdirectory.	If the specified group is already a trusted group, this option has no effect on the diagnostic output or the generated script.
-u <i>user</i>	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 <code>/etc/informix</code> 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.

Element	Purpose	Key Considerations
-n	Directs the onsecurity command to process directories belonging to a system user or system group, such as <code>sys</code> or <code>bin</code> , as not trusted.	
-e	Directs the onsecurity command to not check files in <code>/etc/informix</code> .	
-p	Runs the onsecurity command in a mode that is appropriate for non-root installations.	<p>When you run the command with the <code>-p</code> 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:</p> <ul style="list-style-type: none"> • Processes directories belonging to user and group informix as not trusted. • Excludes files in <code>/etc/informix</code> from the security check.
-G <i>fix action</i>	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 <code>-G</code> option, the command assumes that you intended to specify <code>-G chmod</code> .
-U <i>fix action</i>	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 <code>-U</code> option, the command assumes that you intended to specify <code>-U chown</code> .
-O <i>fix action</i>	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 <code>-O</code> option, the command assumes that you intended to specify <code>-O chmod</code> .
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 <code>-G</code> or <code>-O</code> option is invoked prior.	

Element	Purpose	Key Considerations
add	<ul style="list-style-type: none"> • With -G option: Adds current nonsecure group assigned to directory to the <code>/etc/informix/trusted.gids</code> file • With -U option: Adds current nonsecure owner of directory to the <code>/etc/informix/trusted.uids</code> file • With -O option: Adds nonsecure directories to the <code>etc/informix/trusted.insecure.directories</code> 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 `$INFORMIXDIR` and its subdirectories

When the `$INFORMIXDIR` directory is not secure, run a security script to reset directory permissions.

About this task

To secure the `$INFORMIXDIR` directory and its subdirectories, you can run the `$INFORMIXDIR/etc/make-informixdir-secure` script as user **root** or generate a security script to run with the `onsecurity` utility.

To generate a security script:

1. Run the following command to generate the `secure.sh` script: `$INFORMIXDIR/bin/onsecurity -r $INFORMIXDIR`
2. Run the `secure.sh` script to secure the installation path:

Choose from:

- UNIX and Linux: `$INFORMIXDIR/tmp/secure.sh`
- Mac OS X:
 - a. Open a terminal window.
 - b. Use the `sudo -s` command to acquire user root privileges.
 - c. Go to the directory with the command `cd $INFORMIXDIR/tmp`
 - d. Run `secure.sh`

What to do next

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.

About this task

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.

Result

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.

Results

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-14.10server-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™)

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

Users and group membership for running utilities

Most HCL 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
- 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.

For non-root installations of Informix®, the owner of the installation must also own the chunk files where data is stored. Chunk files for non-root installations of Informix® must have permissions set to 600.

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 HCL® 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 *HCL® Informix® Administrator's Reference*.

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.



Note: Support for Communication Support Module (CSM) is removed starting Informix Server 14.10.xC9. You should use Transport Layer Security (TLS)/Secure Sockets Layer (SSL) instead.

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 `conccsm.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 28](#) 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 HCL® Informix®-specific encryption CSMs. For more information, see [Secure sockets layer protocol on page 29](#).

Enabling encryption with communication support modules

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

Before you begin

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

About this task

To enable network encryption:

1. Add a line to the `conccsm.cfg` file.
The `conccsm.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` information.
For details about specifying the CSM in the `sqlhosts` information, see the *HCL® Informix® Administrator's Guide*.

CSM configuration file

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

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

The `concsn.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 *HCL® Informix® Guide to SQL: Reference*.

Entries in the `concsn.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 21](#).

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 `concsn.cfg` on page 22](#)


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 21](#).

The default MAC key file is the built-in file provided by HCL® Informix®. This file provides limited message verification (some validation of the received message and determination that it has come from the HCL® 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 conccsm.cfg on page 22](#)

Generating a new MAC key file

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

About this task

To generate a new MAC key file:

1. Run the following command from the command line:

Example

```
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:

Choose from:

- **Using encryption tags:** Edit the relevant line in the `conccsm.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 conccsm.cfg on page 22](#)

[ENCCSM_MACFILES parameter on page 27](#)

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 conccsm.cfg on page 22](#)

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 `conccsm.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 conccsm.cfg on page 25](#)
- [Specifying network encryption options in conccsm.cfg on page 22](#)



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

Specifying network encryption options in conccsm.cfg

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

HCL® 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 `conccsm.cfg` file.



Note: Specifying encryption options directly in the `conccsm.cfg` file is usually more difficult than specifying libraries and tags in an encryption parameters file because of syntax specifications. A sample file `conccsm.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/conccsm.cfg` (UNIX™ and Linux™) or `%INFORMIXDIR%\etc\conccsm.cfg` (Windows™).

conscm.cfg entry Syntax

```
(explicit id unique_25_Connect_42_encp002) unique_25_Connect_42_encp002 name(" { client=client_library , server=server_library |
csm_library } " , " [{ config= parameter_file | { <Cipher options> <MAC options> <Switch options> } } ] ")
```

Cipher options

```
" cipher[ "
" { all | allbut: < cipher > | cipher:mode } ] "
```


MAC options

```
" mac[ levels:< { | high | medium | low | off } > "
" files:< { builtin | file_name , [builtin] } > ] "
```

Switch options

```
" switch[ { | { cipher:negotiation_interval } | [key: negotiation_interval] } ] "
```

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 HCL® Informix®. The builtin file provides limited message verification that received messages have come from the HCL 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.

Option	Description
<i>levels</i>	<p>Specifies a comma-separated list of MAC generation levels that the connection supports.</p> <p>high Use SHA1 MAC generation on all messages.</p> <p>medium Use SHA1 MAC generation for all messages greater than 20 bytes long and XOR folding on smaller messages.</p> <p>low Use XOR folding on all messages.</p> <p>off Do not use MAC generation.</p>
<i>mode</i>	<p>Use the specified cipher mode.</p> <p>ecb Electronic Code Book</p> <p>cbc Cipher Block Chaining</p> <p>cfb Cipher Feedback</p> <p>ofb Output Feedback</p>
<i>name</i>	The name that you assign to the CSM.
<i>negotiation_interval</i>	The minutes between renegotiations.
<i>parameter_file</i>	<p>The path and file name of the file in which the encryption parameters are defined.</p> <p> Important: If the file does not exist at the specified path, then default parameter values are used. No error is returned.</p>
<i>server_library</i>	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 reference

[Invoking an encryption parameters file in `concsm.cfg` on page 25](#)

Related information

[MAC key files on page 20](#)

[Encryption ciphers and modes on page 18](#)

[MAC levels on page 21](#)

[Generating a new MAC key file on page 20](#)

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 conccsm.cfg on page 22](#)

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 18](#).

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 21](#).

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 20](#).

Related information

[Generating a new MAC key file on page 20](#)

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 21](#).

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 `conccsm.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.



Note: Support for Communication Support Module (CSM) is removed starting Informix Server 14.10.xC9 . You should use Transport Layer Security (TLS)/Secure Sockets Layer (SSL) instead. Since CSM is not supported starting Informix Server 14.10.xC9, these configuration parameters are also not supported starting Informix Server 14.10.xC9 .

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

Secure sockets layer protocol

Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS), are communication protocols that use encryption to provide privacy and integrity for data communication through a reliable end-to-end secure connection between two points over a network.



Note: In this documentation, the same information applies to TLS as to SSL.

You can use SSL for the following connections:

- HCL Data Server Driver for JDBC and SQLJ connections with Informix®
- HCL Informix® ESQ/C connections with Informix®
- HCL 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 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 SPWDSCM) with SSL connections. However, because these CSMs provide encryption functionality, configuring the ENCCSM or SPWDSCM with SSL involves additional effort with no extra benefit.

You can configure Pluggable Authentication Module (PAM) and the Generic Security Services Communications Support Module (GSSSCM), 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 \(GSKit\) on page 33](#) and <http://www.openssl.org>.

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 algorithm. With a symmetric key algorithm, the same encryption key is used to encrypt and decrypt the data by both communication partners (SSL client and SSL server). As this key can be used to decrypt the data, it must be kept a secret and shared only between the two SSL communication partners. Therefore, at the beginning of an SSL connection an asymmetric key (private-public key) algorithm is used for the exchange of the secret symmetric key.

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 SSL/TLS digital certificate to the client.
2. The client verifies the validity of the SSL/TLS digital certificate sent by the server. For this to occur, the client must possess the root certificate of the CA that issued the SSL/TLS digital certificate to the server.

If the handshake succeeds, these events occur:

1. The client generates a random symmetric key and sends it encrypted to the server. For this encryption, the client uses the public part of the asymmetric key. This public key is contained in the SSL/TLS digital certificate issued to the server.
2. The server receives the encrypted symmetric key and decrypts it with its private key.

Data encrypted with the public key can be decrypted only with the corresponding private key, which is only in the possession of the server. With that it is guaranteed, that the symmetric key remains a secret, shared only between the client and the server.

As both the server and client now know the symmetric key, they can use it to encrypt the data transferred between server and client for the duration of the session. Data encryption and decryption with a symmetric key algorithm is much more efficient

than with an asymmetric key algorithm. Therefore, the asymmetric key algorithm is only used to exchange the symmetric key at the beginning of an SSL connection. Then the symmetric key algorithm is used for the bulk of the data moved during the connection.

In a most simplistic setup for SSL communication, the SSL server only uses a single self-signed certificate. Therefore no involvement of a CA is needed. To authenticate the SSL server, an SSL client needs a copy of the server's self-signed certificate in its own client keystore. During the SSL handshake, the server sends its self-signed certificate to the client. The client looks for the copy of this certificate in its own keystore and compares the two certificates. An exact match is expected for the SSL handshake to succeed.

Encryption modules used to implement SSL communication

To implement the SSL communication, Informix uses one of the two encryption modules, either the IBM Global Security Kit (GSKit) or OpenSSL:

- Until and including version 14.10.xC3, all Informix products use GSKit only. GSKit is packaged and installed with the Informix products themselves.
- With versions 14.10.xC4W1 and newer, GSKit is packaged, installed and used with the Informix Server product. Client products (like Client SDK) that are installed in the same directory as the Informix Server also use the GSKit provided with the Informix Server installation.
- Client products (like Client SDK) of versions 4.50.xC4W1 and newer that are installed in a separate location use OpenSSL.



Note: GSKit is no longer packaged and installed with client products of these versions.

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 a keystore that stores the digital certificates used in SSL communication.

There are two encoding formats for keystores, the open standard PKCS#12 format and a GSKit proprietary format called "CMS". GSKit supports both keystore formats, whereas OpenSSL only supports the PKCS#12 format. By convention, a PKCS#12 format keystore file has the filename extension "*.p12" and a CMS format keystore has "*.kdb" as file name extension. Up to and including version 14.10.xC3, Informix only uses the file name extension "*.kdb" for keystores, and thus propagated the use of the CMS format. With version 14.10.xC4W1 and newer, Informix also supports the file name extension "*.p12". With that we now recommend the use of the standard PKCS#12 format for keystores. Informix product installations that use OpenSSL as encryption module require the PKCS#12 format for their keystores.

The content in a keystore of either format is encrypted and a password is needed in order to access it. Therefore, both, the database server as well as the database client, need the respective password for their individual keystore in order to access it. To allow keystore access without providing the password interactively or upon startup of the database server or the database client, the keystore password can be stored in a password stash file in an encrypted format.

When using GSKit to manage a keystore, the keystore password can be stashed by specifying an extra option to the GSKit utility. Instead, with OpenSSL, you use the new utility "onkstash" to stash the keystore password. This utility "onkstash" is packaged and installed with Informix products of version 14.10.xC4W1 or newer. It accepts the file path name of the keystore and the cleartext password on the command line. With both methods of password stashing, the stash file is created with the same file path name but a different filename extension. The filename extension is either ".sth" or ".stl", depending on which encryption module is used. If GSKit and a GSKit utility is used, the filename extension is ".sth". If OpenSSL and "onkstash" is used, the filename extension is ".stl".

To get the password when accessing their respective keystore, both, the database server as well as the database client, can use the corresponding stash file if it exists at the expected location and has the correct filename extension.

While the content of a standard PKCS#12 keystore itself is independent from the encryption module in use, the password stash file is not compatible between GSKit and OpenSSL. This is because with each encryption module, a specific algorithm is used to encrypt the password in the stash file. Therefore it is important that the correct stash file, i.e. filename extension ".sth" or ".stl", is present for the access to the keystore.

The server keystore and its configuration

The keystore stores the SSL/TLS digital certificate issued to the server and corresponding private key, as well as CA certificates for authenticating all other servers that the Informix® server is connecting to. The server keystore must be located in the directory `$INFORMIXDIR/ssl`. The name of the keystore file must be `server_name.p12` (or `server_name.kdb` with versions 14.10.xC3 or older), 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 must have a unique "label". This label is a name tag that identifies a certificate in a keystore. The term "label" is used by GSKit, whereas in OpenSSL documentation, the same is called "friendly name". When you set up Informix® to use SSL, you must specify the label of the SSL/TLS digital certificate issued to the server with the **SSL_KEYSTORE_LABEL** configuration parameter in the server configuration file.

The keystore is protected by a password that the server must know so that it can access the digital certificates in the keystore for SSL communications. The keystore password is stored in an encrypted form in a stash (`.sth`) file in the directory `$INFORMIXDIR/ssl`. The name of the keystore stash file must be `server_name.sth`. Note the use of the stash file name extension ".sth" as the Informix server uses GSKit as encryption module.

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

The permissions on the `INFORMIXDIR/ssl/server_name.p12` 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 of an Informix® client stores the root certificates needed for the authentication of all servers to which the client is connecting. A database client can retrieve its keystore password from a stash file. This stash file can be created with the

utility **onkstash** (when using OpenSSL) or the utility **gsk8capicmd** (when using GSKit) to contain the keystore password in an encrypted form.

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. Client configuration parameters in the `conssl.cfg` file

HCL® Informix® Client Configuration	
Parameter	Description
SSL_KEYSTORE_FILE	This is the fully qualified file name of the keystore that stores the CA certificates for authenticating of all 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. For versions 14.10.xC4W1 and newer, this parameter is optional and is only needed if the stash file is at a location or has a file name different from the expected default.

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 `$INFORMIXDIR/etc/client.p12` (or `$INFORMIXDIR/etc/client.kdb` for versions 14.10.xC3 and older) and `$INFORMIXDIR/etc/client.sth` (with GSKit) or `$INFORMIXDIR/etc/client.stl` (with OpenSSL) as the default keystore and keystore stash file names.

Related information

[GSKIT_VERSION configuration parameter on page](#)

[The onkstash Utility on page](#)

IBM® Global Security Kit (GSKit)

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

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

IBM® Global Security Kit version 8 is installed with Informix® 14.10 and HCL® Informix® Client Software Development Kit (Client SDK) version 4.10. IBM® Global Security Kit 8 is used by default; however, if you have another supported, major version of IBM® Global Security Kit software installed on your computer, you can set the `GSKIT_VERSION` configuration parameter so that the database server uses the other version.

The IBM® Global Security Kit 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* ftp://ftp.software.ibm.com/software/webserver/appserv/library/v80/GSK_CapiCmd_UserGuide.pdf.



Note: Follow your company policies to obtain certificates.

Configuring a server instance for secure sockets layer connections

Configure the HCL 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.

About this task



Note: Transport Layer Security (TLS) is the successor to SSL. In this documentation, the same information applies to TLS as to SSL.

To configure the 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:

Choose from:

- `onsocssl` protocol for ESQL/C, ODBC, DB-Access, `dbexport` utility, `dbimport` utility, `dbschema` utility, or `dbload` utility connections
- `drsocssl` protocol for DRDA® connections

Example

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

Table 3. Example of `sqlhosts` file configured for SSL connections

Server Name	Protocol	Host Name	Service Name
server1_on	onsoctcp	sanfranci	s1_on
		sco	
server1_on_ssl	onsocssl	sanfranci	s1_on_ssl
		sco	
server1_dr_ssl	drsocssl	sanfranci	s1_dr_ssl
		sco	

For more information about the `sqlhosts` file or SQLHOSTS registry, see the *HCL® Informix® Administrator's Guide*.

2. Update configuration parameters in the `onconfig` file, as follows:

- a. Specify the label of the SSL/TLS digital certificate issued to the server in the **SSL_KEYSTORE_LABEL** configuration parameter.

The label can contain up to 512 bytes.

For example, specify:

```
SSL_KEYSTORE_LABEL server1_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 **iiipp**, 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.

- d. If you want to control the version of the TLS protocol to be used, set the configuration parameter **TLS_VERSION** accordingly.

For example, specify:

```
TLS_VERSION 1.2,1.3
```

to allow the use of TLS protocol versions 1.2, 1.3 .



Note: TLS versions 1.0 and 1.1 are disabled starting Informix Server version 14.10.xC8 .

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, a supported Java™ runtime environment must be installed. The GSKCapiCmd tool is part of the GSKit and does not require Java™.

When you create the keystore, be sure to:

- Select the option to stash the password.
- Name the keystore as `servername.p12` (versions 14.10.xC4W1 and newer) or `servername.kdb` (versions 14.10.xC3 and older), where `servername` is the 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/servername.p12` (or `$INFORMIXDIR/ssl/servername.kdb`) file and the `$INFORMIXDIR/ssl/servername.sth` file to `600`, with `informix` set as both the owner and the group, even though Informix® does not enforce these permissions.

For example with versions 14.10.xC4W1 and newer, use:

```
gsk8capicmd -keydb -create -db server1.p12 -pw server1_password -stash
gsk8capicmd -cert -create -db server1.p12 -stashed -label server1_ssl -dn 'CN=Informix Server 1' -size
4096 { -ca true -sigalg
SHA256WithRSA }
```

For example with versions 14.10.xC3 and older, use:

```
gsk8capicmd -keydb -create -db server1.kdb -pw server1_password -stash
gsk8capicmd -cert -create -db server1.kdb -stashed -label server1_ssl -dn 'CN=Informix Server 1' -size
4096 [ -ca true -sigalg
SHA256WithRSA ]
```



Note:

- For 64bit products, you need to add `_64` to the command name i.e. use `gsk8capicmd_64`.
- The above commands create a simple keystore that contains a single self-signed certificate and a key. As such, the certificate is the SSL/TLS digital certificate issued to the server and at the same time is to be used by the client in lieu of a CA certificate. In such a scenario, there is no certificate from a CA involved.
- While it is possible to continue using CMS format keystores (`“.kdb?”` file name extension) with versions 14.10.xC4W1 and newer, we recommend the use of the PKCS#12 open standard format for keystores (file name extension `“.p12?”`).
- Versions 14.10.xC3 and older depend on the file name extension for keystores being `“.kdb?”`.
- The first command of the examples creates an empty keystore. The given option `“-stash?”` causes the password to be stashed in the corresponding file `server1.sth`. With the password already stashed, subsequent commands can be given the option `“stashed?”` instead of having to repeat the password. The second command creates a self-signed certificate and a key in the keystore.
- In the example, the option `“-label server1_ssl?”` specifies `“server1_ssl?”` as an identifier of the certificate created in the keystore. This is the value that must correspond to the setting of parameter `SSL_KEystore_LABEL` in the `onconfig` file.



Tip: You can find the labels of certificates in a keystore with a command like:

```
gsk8capicmd -cert -list -db server1.p12 -stashed
```

- In the example, the option `“-dn ‘CN=Informix Server 1’?”` specifies the distinguished name of the certificate with just the common name being `“Informix Server 1?”`.



- At least with versions 14.10.xC4W1 and newer, we recommend the use of options “-ca true?” and “-sigalg SHA256WithRSA?” when creating a (self-signed) certificate. These options make sure that certificates are created and signed with certain attributes that may be required by future versions of encryption libraries, especially in the case of TLS 1.3 .



Tip: To get detailed information about a specific certificate in a keystore, use a command like the following, specifying the label of the certificate:

```
gsk8capiCmd -cert -details -db server1.p12 -stashed -label server1_ssl
```

- If you are planning to use TLS 1.3, -size argument should have the value at least 4096.
- Refer *GSKCapiCmd User's Guide* ftp://ftp.software.ibm.com/software/webserver/appserv/library/v80/GSK_CapiCmd_UserGuide.pdf for further details.

For information about the keystore, the password stash file, and digital certifications, see [Secure sockets layer protocol on page 29](#).

For information about the iKeyman utility, GSKCmd command-line interface, and the GSKCapiCmd command-line interface, see [IBM Global Security Kit \(GSKit\) on page 33](#).

What to do next

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 on page 37](#).

Related information

[GSKIT_VERSION configuration parameter on page](#)

Configuring a client for SSL connections

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

Before you begin

The client and the server must be enabled with a mutual TLS version. Set the TLS version on the server with the `TLS_VERSION` configuration parameter.

About this task



Note: Transport Layer Security (TLS) is the successor to SSL. In this documentation, the same information applies to TLS as to SSL.

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

Example

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

Table 4. Example of `sqlhosts` file configured for SSL

SQLI client connections

Server Name	Protocol	Host Name	Service Name
sf_on	onsoctcp	sanfranci sco	sf_serv
oak_on	onsocssl	oakland	oak_serv

2. Using a text editor, create a `conssl.cfg` file in the `$INFORMIXDIR/etc` directory. The file must contain the following information:

Choose from:

- **SSL_KEYSTORE_FILE** 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** that specifies the fully qualified file name of the keystore password stash file. With versions 14.10.xC4W1 and newer, this parameter is optional and needed only if the stash file is at a different location or has a different file name than the keystore file itself.

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.p12 # Keystore file
```

3. Create the keystore.

The client keystore must contain all the certificates that are needed to authenticate the server during the SSL handshake. Generally, these certificates are CA certificates. You get such CA certificates from the CA that signed the server's certificate. However, if the server's certificate is self-signed, then the client keystore must contain a copy of this self-signed certificate in its keystore. In this case you must extract that certificate from the server keystore and add it to the client keystore.



Note: Follow your company policies to obtain the server certificates.

Client versions 4.50.xC3 and older use GSKit as encryption module. Therefore, with these versions it is possible to use the GSKit proprietary CMS keystore format. However, we recommend to use the standard PKCS#12 format with

file name extension ".p12" for client keystores. A separately from a server installed stand-alone client of version 14.10.xC4W1 or newer uses OpenSSL as encryption module. For these installations the keystore must be in the standard PKCS#12 format.

When you create the keystore, be sure that:

- The name and location of the keystore and optional stash file are as specified in the `conssl.cfg` file as explained in step 2 above.
- Permissions on the keystore and its stash file are set to 666, even though the permissions are not enforced.

Use the external utilities from GSKit or OpenSSL to create the client keystore. Following are the steps to create the keystore:

3.1 Using OpenSSL:

For more information on the OpenSSL utility commands, see the OpenSSL documentation at <http://www.openssl.org>.

- If the certificate created for the server is self-signed,
 - a. Log on to the database server machine and extract the certificate from the server keystore:

```
-openssl pkcs12 -in $INFORMIXSERVER.p12 \  
-passin pass:SERVERPASSWD \  
-out SSL_KEystore_LABEL.cert.pem -nokeys
```



Note: If the format of the server's keystore is CMS (filename extension ".kdb?"), then you need to use a GSKit utility like `gsk8capicmd` to extract the certificate from this keystore. (The CMS keystore format is a GSKit proprietary format.) See below for a `gsk8capicmd` example to extract the server's certificate from the keystore.



Tip: If you don't know the password of the server's keystore but the password is stashed, you can use the GSKit `gsk8capicmd` utility to extract the certificate without specifying the keystore password. See below for a `gsk8capicmd` example.

- b. Transfer the output file `SSL_KEystore_LABEL.cert.pem` from the server machine to the client machine.
- c. Create the client keystore using the exported certificate in the PEM file as input:

```
openssl pkcs12 -export -out client.p12 -passout pass:CLIENTPASSWD \  
-in SSL_KEystore_LABEL.cert.pem -caname LABEL -nokeys
```



Note: You must specify the option "-caname ..." in order to give a label ("friendly name?") as identifier to the certificate in the client keystore.

- d. Use the utility `onkstash` to create the stash file with the keystore password:

```
onkstash client.p12 CLIENTPASSWD
```

- If your client connects to several different database servers or your database server has a CA-signed certificate:
 - a. Collect all the certificates needed for the authentication of each database server into a single PEM file:
 - For a database server that uses a single self-signed certificate, use a command as described above to extract that certificate from the server's keystore
 - For a database server that uses a certificate signed by a CA, get all needed CA certificates from the CA
 - In case there are duplicates in the collected certificates, eliminate all duplicates. Duplicate certificates may be collected e.g. when the same CA signed the certificates of several different database servers
 - Concatenate all unique certificates into a single PEM file
 - b. On the client machine, create the client keystore using the single PEM file with the unique certificates as input:

```
openssl pkcs12 -export -out client.p12 -passout pass:CLIENTPASSWD \
-in SINGLE_PEM_FILE -nokeys -caname LABEL1 -caname LABEL2..
```



Note: You must specify the option “-caname ...?” multiple times with an individual label name as unique identifier for each certificate in the input PEM file.

- c. Use the utility onkstash to create the stash file with the keystore password:

```
onkstash client.p12 CLIENTPASSWD
```

3.2 Using GSKit:

Use the GSKCapiCmd command-line interface to set up a keystore and its password stash file and digital certificate.



Note:

- If the password of the keystore is stashed, you may replace the option “-pw <passwd>?” with the option “-stashed?”.
- Replace file extension “.p12?” with “.kdb?” for versions 14.10.XC3 and older OR if the format of the server's keystore is CMS instead of PKSC#12.

For example:

- Run the following command on the client machine to create an empty PKCS#12 keystore and stash the password:

```
GSK_COMMAND -keydb -create -db client.p12 -pw CLIENTPASSWD -stash
```

For database clients of versions 14.10.XC3 and older, instead use the following command to create an empty CMS format keystore and stash the password:

```
GSK_COMMAND -keydb -create -db client.kdb -pw CLIENTPASSWD -stash
```

- Get the certificate(s) needed to authenticate the server:

- If the server's certificate is self-signed:
 - a. Log on to the remote server and extract the certificate from the server keystore:

```
GSK_COMMAND -cert -extract -db $INFORMIXSERVER.p12 -pw SERVERPASSWD -label
SSL_KEYSTORE_LABEL -target SSL_KEYSTORE_LABEL.cert.pem -format ascii
```

where, SSL_KEYSTORE_LABEL stands for the label name of the server's certificate in the server's keystore, This is the value that is also specified for the server's configuration parameter SSL_KEYSTORE_LABEL.

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

- If the server's certificate is signed by a CA:
 - Get all the needed CA certificates from the CA
- Add each certificate to the client keystore by running a command :

```
GSK_COMMAND -cert -add -db client.p12 -stashed -label
LABEL1 -file SSL_KEYSTORE_LABEL.cert.pem -format ascii
```



Note: You must specify a unique label name as identifier for each certificate that you add to the client keystore.

- If your client connects to several different database servers:
 - Repeat the above steps for each database server



Note: For the migration of pre-existing client keystores used with GSKit, see [Using SSL/TLS database connections on page](#) in the Migration Guide.

Related information

[GSKIT_VERSION configuration parameter on page](#)

[TLS_VERSION configuration parameter on page](#)

Renewing expired certificates in a keystore

A certificate in a keystore is valid for a limited period of time, after which the certificate expires. Once the certificate expires, it can cause the client-server communication to fail. It is important to renew the certificate.

Renewing a CA-signed certificate

About this task

If a CA signed certificate is close to expiry, it is possible to renew the certificate by recreating a new certificate signing request.

In this task, as an example, refer **cert1_renew.csr** as the recreated certificate request and **cert1_renew.pem** as the renewed certificate signed by the CA.

1. List the certificates in the keystore, then use the label "cert1" of the personal certificate to get detailed information on this certificate. Check the expiration date shown as the "Not after:" value.

```
gsk8capicmd_64 -cert -list -db renew.p12 -stashed
Certificates found

* default, - personal, ! trusted, # secret key

!      ddRoot
-      cert1
gsk8capicmd_64 -cert -details -db renew.p12 -stashed -label cert1
Label : cert1
Key Size : 2048
Version : X509 V3
Serial : 6363d2bef7cd7e20
Issuer : CN=Dave,C=US
Subject : CN=domain,O=you,C=US
Not Before : September 28, 2022 4:29:52 PM CDT

Not After : September 30, 2022 4:29:52 PM CDT
```

2. Recreate the certificate signing request by running:

```
gsk8capicmd_64 -certreq -recreate -db renew.p12 -stashed -label cert1 -target cert1_renew.csr
```

3. Send the newly recreated certificate request to the original Certificate Authority (CA) for signing. Upon receiving the renewed certificate from the CA in file "cert1_renew.pem", place this renewed certificate in your server keystore by running:

```
gsk8capicmd_64 -cert -receive -db renew.p12 -stashed -file cert1_renew.pem
```

4. Verify the details of the renewed certificate:

```
gsk8capicmd_64 -cert -details -db renew.p12 -stashed -label cert1

Label : cert1
Key Size : 2048
Version : X509 V3
Serial : 303c392837085051
Issuer : CN=Dave,C=US
Subject : CN=domain,O=you,C=US
Not Before : September 28, 2022 4:45:04 PM CDT

Not After : September 24, 2023 4:45:04 PM CDT
```

Renewing a self-signed certificate

About this task

If a self-signed certificate is about to expire, it is possible to renew it. You do not have to create a new certificate.

1. List the certificates in the keystore and use the label "cert1" of the personal certificate to get detailed information on this certificate. The value of "Not After:" shows the certificate's expiration date.

```

gsk8capicmd_64 -cert -list -db ssign.p12 -stashed
Certificates found

    default, - personal, ! trusted, # secret key
    -      cert1
gsk8capicmd_64 -cert -details -db ssign.p12 -stashed -label cert1
Label : cert1
Key Size : 2048
Version : X509 V3
Serial : 450dfa054a483e88
Issuer : CN=domain,O=you,C=US
Subject : CN=domain,O=you,C=US
Not Before : September 28, 2022 5:00:27 PM CDT

Not After : September 30, 2022 5:00:27 PM CDT

```

2. Renew the certificate for a period of one year:

```

gsk8capicmd_64 -certreq -recreate -db ssign.p12 -stashed -label cert1 -target ssign.csr
gsk8capicmd_64 -cert -sign -db ssign.p12 -stashed -label cert1 -file ssign.csr -target ssign.pem -expire
360
gsk8capicmd_64 -cert -receive -db ssign.p12 -stashed -file ssign.pem
gsk8capicmd_64 -cert -details -db ssign.p12 -stashed -label cert1

Label : cert1
Key Size : 2048
Version : X509 V3
Serial : 6c240165e3561ddb
Issuer : CN=domain,O=you,C=US
Subject : CN=domain,O=you,C=US
Not Before : September 28, 2022 5:09:12 PM CDT

Not After : September 24, 2023 5:09:12 PM CDT

```

Background Knowledge on Keystores

This topic offers some generic insights into keystores and how they are used for secure communications with the TLS (Transport Layer Security) protocol. While the first part provides the theoretical background, the second part shows examples for applying this in practice using OpenSSL.

Concepts of Keys, Certificates and Keystores for TLS

This section explains keystores for TLS/SSL (Transport Layer Security / Secure Socket Layer) connections between database clients and servers.

With the client-server architecture being the norm for database systems, the communication between the database client and the database server generally occurs over a network connection. While such a network connection may be completely inside a private network, like an intranet of a company, it can just as well include public sections using the internet. Especially connecting to the ever more popular servers and services in the cloud almost always involves sections of the public internet. In addition, new legislation, like the GDPR (General Data Protection Regulation) in the European Union, require a stronger protection for the privacy of data. Securing the communication between database client and server is an important aspect and a requirement to be taken seriously.

A high level view of TLS/SSL

TLS and its meanwhile deprecated predecessor SSL are an evolving series of protocol versions that use various encryption algorithms to secure an individual connection between a client and a server. With regard to the OSI (Open Systems Interconnection) model, the TLS/SSL protocols are situated in the presentation layer. An important part of the protocol is the initial negotiation of encryption algorithms and methods that are to be used for the connection. This is also known as the SSL/TLS handshake. It allows a client and server with different capabilities to figure out, which protocol version, algorithms and methods to use in order to best comply with the given requirements.

The authentication of the communication partner during the TLS handshake is optional. Most commonly, only the client authenticates the server it connects to. With that, the client as connection initiator can rest assured to connect to the desired server and not to an imposter. Rarely, the server also authenticates the client as part of the TLS handshake. Often, client authentication is done by other means when the connection already is established and secured, for example, by requiring the user to provide some kind of user ID and password.

To encrypt the data transferred over a secure connection, the client and server must agree on an encryption key that they can use for the duration of the connection. As symmetric encryption is used, both communication partners must have the same encryption key and use it to encrypt data before sending it, as well as to decrypt received data. Obviously, this encryption key must remain a secret, shared only by the client and the server. Anybody else knowing this encryption key would be able to eavesdrop on the communication, decrypt the data and thus be able to read the clear text. Therefore, this agreement on a symmetric encryption key, only shared between client and server, must be done in a secure way - before the communication data can be encrypted.

This can be done by using asymmetric encryption for the transfer of the secret symmetric encryption key during the TLS handshake. Asymmetric encryption uses a pair of encryption keys. The public key is used to encrypt data. This data can be decrypted only with the corresponding private key of the pair. Usually, the server owns both keys and sends the public key to the client. As this key is not a secret, it can be sent without encrypting it. The client generates a random key as the symmetric key to be used for the connection. It encrypts this symmetric key using the public key received from the server and then sends this encrypted symmetric key to the server. The server decrypts the symmetric key using the private key. With that, both communication partners now are in possession of the same symmetric key and can subsequently use it to encrypt all the data sent over the connection. As only the server possesses the private key, nobody apart from the server can decrypt the encrypted symmetric key sent from the client to the server. With that, the transfer of the symmetric key from the client to the server is secure.

The symmetric encryption is computationally much more efficient than asymmetric encryption. Therefore, the asymmetric encryption is only used for the secure transfer of the relatively small symmetric key itself during the TLS handshake. Then symmetric encryption can be used for the bulk of the data transferred between client and server for the duration of the connection

TLS handshake, certificates and public-private key pairs

TLS uses certificates of the X.509 v3 certificate standard for the authentication of the communication partner. The structure of such certificates is expressed in ASN.1 (Abstract Syntax Notation One) and therefore is very flexible. For a basic understanding of the authentication during the TLS handshake, only a few certificate components are considered here: the

issuer name, the subject name, the subject public key and the certificate signature. Besides these, certificates have a defined begin and end date of their validity, a serial and a version number, among other components. The issuer and subject names are distinguished names that commonly contain fields like country, organization, organizational unit, distinguished name qualifier, state or province name, common name and a serial number. Less often used fields in a distinguished name are a locality, title, surname, given name, initials, pseudonyms and a generation qualifier. In some cases, an e-mail address is included in the common name field.

The issuer name is the distinguished name of the certificate issuer. Usually, the issuer of a certificate is a certificate authority (CA), an entity that is trusted by all communication partners. When issuing (or "signing") a certificate, the CA creates the certificate signature using its own private key and thereby confirms the validity of the certificate. In particular, this means that the subject public key of the certificate is bound to the subject name. The validity of the certificate signature in turn can be verified by using the public key of the CA. The CA's public key is contained in the CA's own certificate, the so called CA certificate.

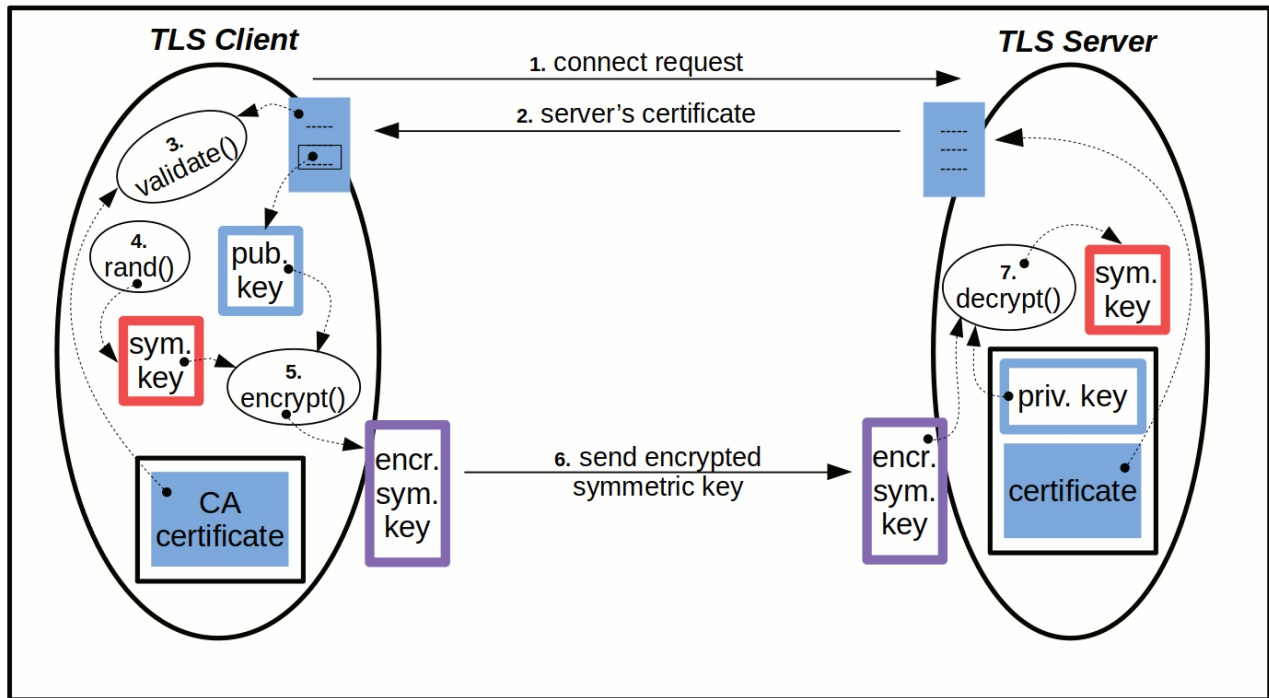
The subject name is the distinguished name of the certificate owner, i.e. the person or entity that receives the issued certificate from the CA. Additional verification checks may be done by a CA to verify that the certificate owner really corresponds to the subject name. The subject public key is the public key for which the certificate owner possesses the matching private key.

When a client authenticates a server, the client has already received the server's certificate. In this certificate, the client finds the issuer name, where the issuer normally is a CA. Using the issuer name, the client checks whether it has its own copy of this issuer's CA certificate. The client (hopefully) finds a copy of this CA certificate and uses the CA's public key in this CA certificate to verify the certificate signature in the server's certificate. As the client itself has its own copy of the CA certificate, the client trusts that this really is not a manipulated copy of the original CA certificate. And the client trusts the CA, i.e. that the CA issued and signed the server's certificate for this server (and not somebody else). With that, the client rests assured, that the server's certificate received from the server really is from this desired server (and not from an imposter). The client now has successfully authenticated the server and proceeds with the TLS handshake.

At this point, you may note that anyone could have previously received the server's certificate (or picked it up by eavesdropping) and use it to pose as the server. However, continuing with the TLS handshake, the client uses the public key in the server's certificate to encrypt the randomly created symmetric key for the connection. An imposter could receive this encrypted symmetric key, but would not be able to decrypt it, because only the real server possesses the private key needed for the decryption. Therefore, a wannabe imposter, who only somehow obtained the server's certificate, cannot successfully establish a secure connection with the client.

As we have just seen, the server's certificate received by the client serves two purposes: For one, the client uses it to authenticate the server. But as the certificate also contains the server's public key, the client can extract it from the certificate and use it right away to encrypt the randomly created symmetric key before sending it to the server.

Figure 2. Simplified TLS handshake



1. The TLS client sends a connect request to the TLS server.
2. The TLS server sends its own certificate to the TLS client.
3. The TLS client uses its own copy of the CA certificate to validate the server's certificate, i.e. authenticate the server.
4. The TLS client generates a random symmetric key.
5. The TLS client uses the public key from the server's certificate to encrypt the generated symmetric key.
6. The TLS client sends the encrypted symmetric key to the TLS server.
7. The TLS server uses its own private key to decrypt the received encrypted symmetric key.
8. Now both, TLS client and TLS server, have the same symmetric key and can start using it to encrypt and decrypt communication data.

Certificates and the corresponding keys play an important role in the TLS handshake. They need to be readily available for use during the TLS handshake. But at the same time they should be sufficiently protected. The server wants to keep its private key out of sight from everybody else. The server and the client also don't want anybody to manipulate the certificates. Especially the client wants to be sure that the CA certificates used to authenticate servers are authentic copies of the CA's original certificates rather than fake certificates planted by somebody else to falsely authenticate some fake server certificate. Therefore, both, the server as well as the client, use keystores to keep their key and certificates safe and accessible at the same time.

Relationship between certificates and keys

For authentication purposes as in the TLS handshake, certificates and keys are pairs. The private key is kept separate, as it must remain private to its owner. The corresponding certificate not only contains information about the owner (subject) and

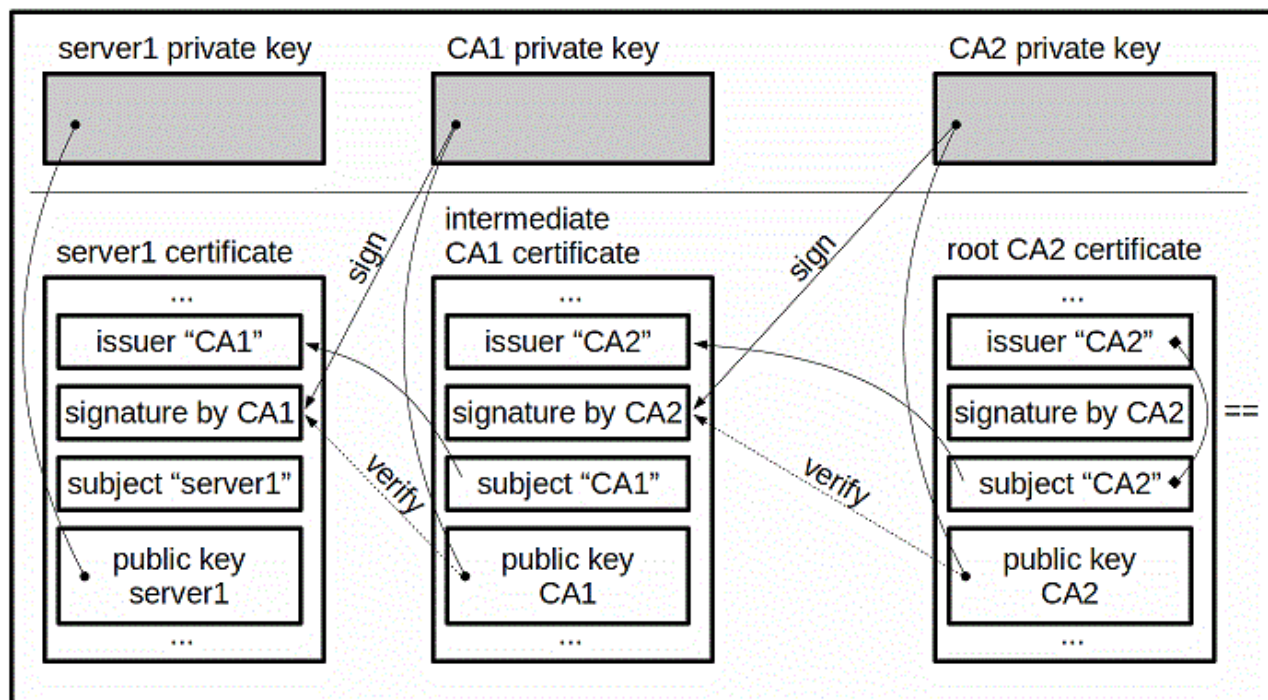
issuer (as distinguished names), but also the public part of the private-public key pair, as well as a signature. The signature in the certificate is created by the issuer of the certificate. To create the signature, the issuer must use its own private key. This signature then can be verified using the public key of the issuer.

Normally, the server uses a certificate that is issued by a trusted third party, a certificate authority (CA). The certificate used by the server therefore is called a user certificate, or also server certificate. The CA certificate is the certificate of the CA that issued and signed the server certificate. The CA certificate contains the CA's public key needed by the client to verify the CA's signature in the server certificate. With that, the server owns its private key and the server certificate, whereas the client only possesses a copy of the CA certificate.

If the CA is trusted by all parties, then the CA certificate is issued and signed by the CA itself. In such a certificate the issuer name and the subject name are the same, and the certificate is called a root CA certificate. The client therefore only needs this root CA certificate to authenticate a server.

In some cases, a client may not directly trust the CA that issued a server certificate. In this case, the client possesses a copy of the CA certificate that in turn is issued and signed by yet another CA. The client sees that in the CA certificate the issuer and subject names are different. As the client does not trust this intermediate CA, the client also needs to find a root CA certificate that issued and signed the intermediate CA certificate. It becomes obvious, that this can recursively repeat itself for a couple of times, until the client finds a trusted root CA certificate. The certificates involved therefore build a so called certificate chain, with the server certificate at one end, the root CA certificate at the opposite end, and a varying number of intermediate CA certificates in the middle.

Figure 3. Simplified 3-element certificate chain



It is possible, that a server itself issues its own server certificate. Such a certificate is called a selfsigned (server) certificate. In this case, there is no CA involved, and the client does not need any CA certificate to authenticate the server. Instead, the client possesses an exact copy of the server's selfsigned certificate (received by "other means"). To authenticate a server, the client checks that the copy of the self-signed certificate in its own possession exactly matches the server certificate received during the TLS handshake.

Keystores

The server as well as the client have reasons to keep their key and certificates needed for TLS communications in a safe place. The server mainly wants to make sure that its private key remains private. Whereas the client is concerned with the certificates being authentic, i.e. that they have not been manipulated. For both purposes a keystore is used to keep encryption keys and certificates in a safe and organized place. As an abstract concept, a keystore can be implemented in different ways: as a collection of files in a directory structure with restrictive access permissions, in some proprietary format of a "database file" or in a file with a standardized format. Keeping so called "PEM" files in a directory is an example for the first kind of keystores and can be used with OpenSSL. The proprietary "CMS" format of "*.kdb" files implemented by IBM's GSKit is an example for the second kind. This section concentrates on the public PKCS #12 standard for keystore files, an example for the third implementation kind.

PKCS #12 defines a file format for storing different cryptography objects in a single file. It is part of the Public-Key Cryptography Standards (PKCS) family of standards published by RSA Laboratories. PKCS #12 is the successor to Microsoft's "PFX" and therefore offers some compatibility between these two formats.

In addition to (local) file keystores, there are also several remote keystore services. Proprietary examples of these are Amazon Web Services Key Management Service (AWS KMS) and Microsoft Azure Key Vault. Key Management Interoperability Protocol (KMIP) services are available from several vendors or providers and adhere to the public KMIP standard.

Keystore content

Cryptography objects like a private key and certificates are stored in individual SafeBag containers within a PKCS #12 keystore file. Normally, at least the SafeBag containing a private key is encrypted using a PBE (password based encryption) method. The keystore file itself also can be encrypted, usually with the same password. With that, access to all content in a PKCS #12 keystore is secured with a password.

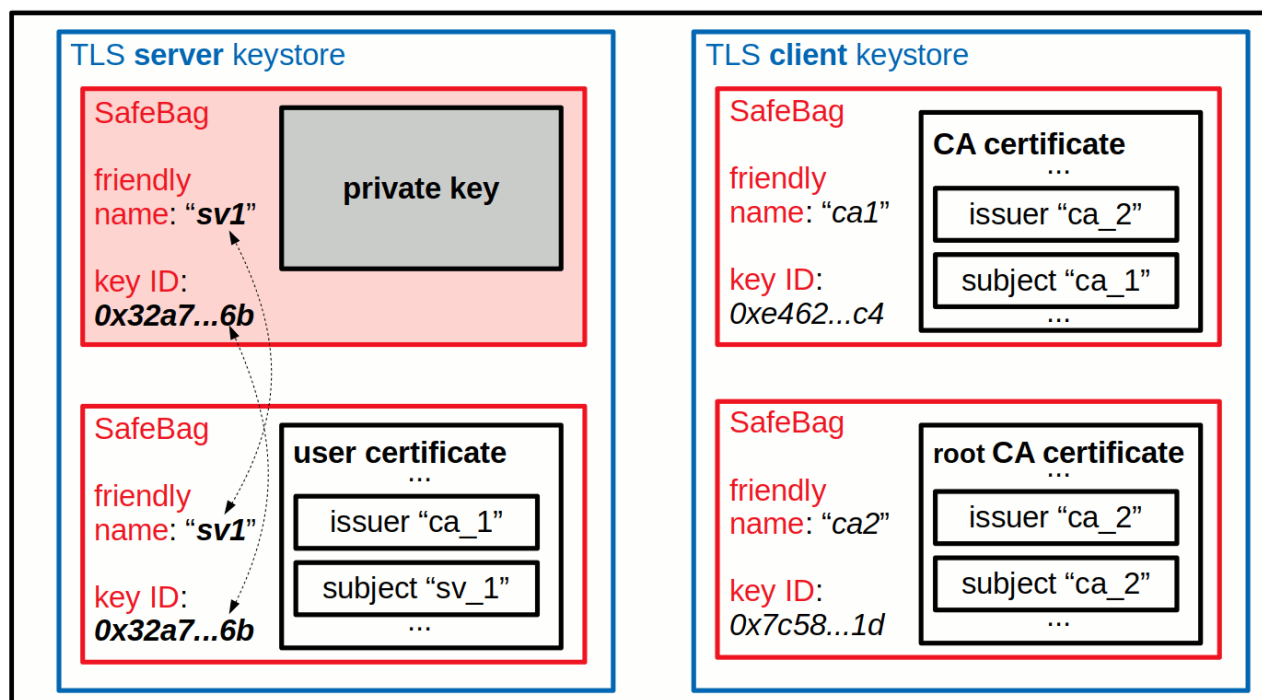
The Safebag containers can have several attributes. The most commonly used SafeBag attributes are a Friendly Name and a Key ID. These are used to identify the objects in the keystore and relate them to each other, as explained below. Especially the more human readable Friendly Name is useful for managing the objects in a keystore.

For TLS communications, the keystore of a server contains just one private key. And for a private key in a keystore, there also must be the corresponding user certificate with the public key of the key pair. The private key and the corresponding user certificate are stored in different SafeBags, because at least the SafeBag with the private key is encrypted. As the certificate is used publicly, the SafeBag containing it normally is not encrypted. The relation between the private key and the corresponding user certificate in their individual SafeBags is realized by both SafeBags having the same Friendly Name

and the same Key ID attributes. The Friendly Name makes it also easy for an administrator to understand, which is the user certificate corresponding to the private key in a keystore.

Besides the private key and the corresponding user certificate, the keystore may contain additional certificates. These are then considered CA certificates. They could be the CA certificates from the certificate chain used for signing the server's user certificate, but could just as well be completely unrelated CA certificates. For a server it is normally sufficient to have its private key and the corresponding user certificate in the keystore. CA certificates can be present, but are not needed, because the server normally does not authenticate the client. For a client, the CA certificate(s) needed to authenticate the server must be in the keystore. The client does not need a private key and therefore also has no user certificate.

Figure 4. Server and client keystore example



Client authentication during the TLS handshake is a seldom used option of the TLS protocol. If activated, the server asks the client for its own user certificate, sometimes also called client certificate. In this case, the client has its own private key and the corresponding user certificate in its keystore, and sends this user certificate to the server to fulfill the server's request. The server in turn needs the proper CA certificate(s) in its own keystore, so that it can use them to authenticate the client. These CA certificates are from the certificate chain that was used to issue and sign the client's user certificate. Often, both user certificates, the server's as well as the client's, are issued and signed by the same CA(s). Then the CA certificates in both keystores are the same. However, this is not a requirement. The user certificates of server and client can just as well be issued by different CAs.

Certificate authorities

As explained before, a certificate authority (CA) is "an entity that is trusted by all communication partners". For TLS communication, this means that the CA is trusted to issue a user certificate only after having checked that the requester for the certificate really is who he says.

For TLS communication, CA is trusted to issue a user certificate only after having checked that the requester for the certificate really is who he says. The extent of this checking is of various degrees. Basic checking usually means verifying that the requester is the owner of the stated e-mail address and internet domain. More elaborate verification may include a proper postal address, entry in a trade register, and even more background checks. The more checking is performed, the trustworthier are the CA and the certificates it issues. These factors also have an effect on the price to be paid for the certificate. This applies mostly to public CAs.

Returning to the basic principle of "an entity that is trusted by all communication partners", a company or organization also can setup its own CA for all internal database communication. As long as all database servers and database clients reside e.g. within the same intranet, all in-house communication partners can agree upon and trust a "home-grown" CA that issues user certificates for the database servers and distributes its own CA certificates to the database clients. As a concept, this may even be acceptable for certain B2B communications, where a database client may be outside of the intranet. Still, the client may trust the CA certificate received from the company for access to a database service of this same company.

With that, the use of CAs can be divided into three basic categories:

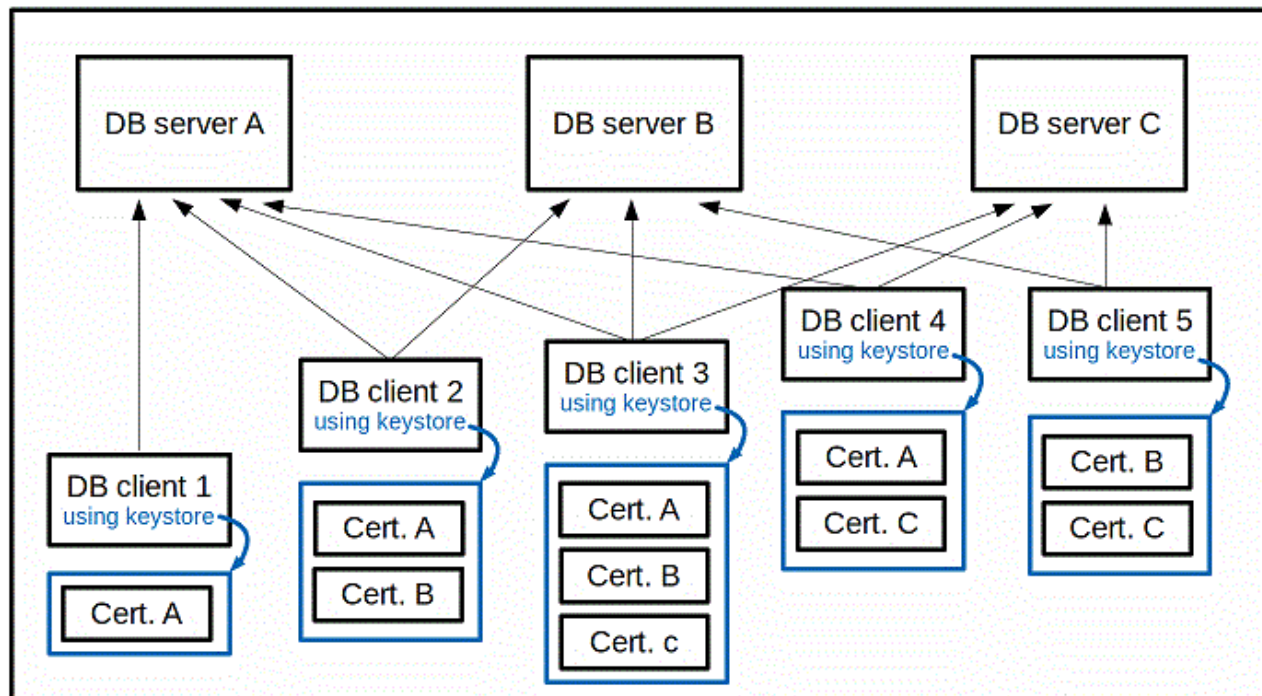
- Using a public CA (and paying for the certificates issued).
- Setting up a home-grown CA (for a limited number of well known and trusting communication partners).
- Not using a CA at all (but only using self-signed certificates).

Using only self-signed certificates

This is the simplest setup for TLS communications. The server only needs a private key and the corresponding self-signed certificate that contains the public key of the key pair. The client only needs an exact copy of the server's self-signed certificate. No CA certificate is needed. Therefore, it is not necessary to request a certificate from a public CA and pay for it, nor is it necessary to set up a home-grown CA to issue certificates for the server. The client receives the copy of the self-signed certificate from a trusted source, probably some system administrator for the in-house database environment. With that, such a setup is quite valid from a security point of view. And it is simple to implement, at least as long as clients connect to just one single SSL server.

However, as such a simple database environment grows over time and more database servers get added, things become more complex. A client that connects to more than one database server needs an exact copy of each individual server's self-signed certificate. Managing the database clients' keystores gets more complicated, as the keystores contain multiple self-signed certificates, probably with different expiration dates. If one of the certificates expires, it may be necessary to re-create the complete keystore, combining the still valid certificates with the new replacement for the one expired certificate. The same is necessary when a new database server gets added to the environment. Every database client that should connect to this new database server needs its existing keystore to be re-created by adding the new self-signed certificate of the new database server. In a rather dynamic environment, this can become an ongoing and quite tiresome task.

Figure 5. Self-signed certificates in client keystores for a 3-server environment



One "simple solution" in such an environment would be to give each database server the same self-signed certificate with the same private key. That way, all database clients only need a copy of this one self-signed certificate and can use it to authenticate all database servers alike when connecting to them. The private key in such a scenario would no longer be very private. Most probably, any responsible security administrator will veto such an approach. A second "simple solution" might be to give each database client the same keystore with all the self-signed certificates of all the database servers, regardless of whether an individual database client needs all of them or not. Still, changes in the database server landscape require updating of all database client keystores.

Using a CA is far easier in a dynamic environment with more frequent changes in the database server landscape.

Setting up a home-grown certificate authority (CA)

While it seems easier to set up keystores for TLS communications without involving a CA, this is not always the case. Especially in an environment with a dynamic database server landscape, properly maintaining the keystores for the database clients with all the self-signed certificates can become a permanent headache. On the other hand, using certificates from a public CA (instead of only self-signed certificates) entails the overhead of dealing with the public CA and the associated costs. Therefore, setting up a "home-grown" CA can be a valid compromise. As long as all communication partners (e.g. within a company intranet) agree to trust their own CA, there may be little reason to use a publicly trusted CA. Instead, the complete control over one's own CA and the absence of costs are strong advantages.

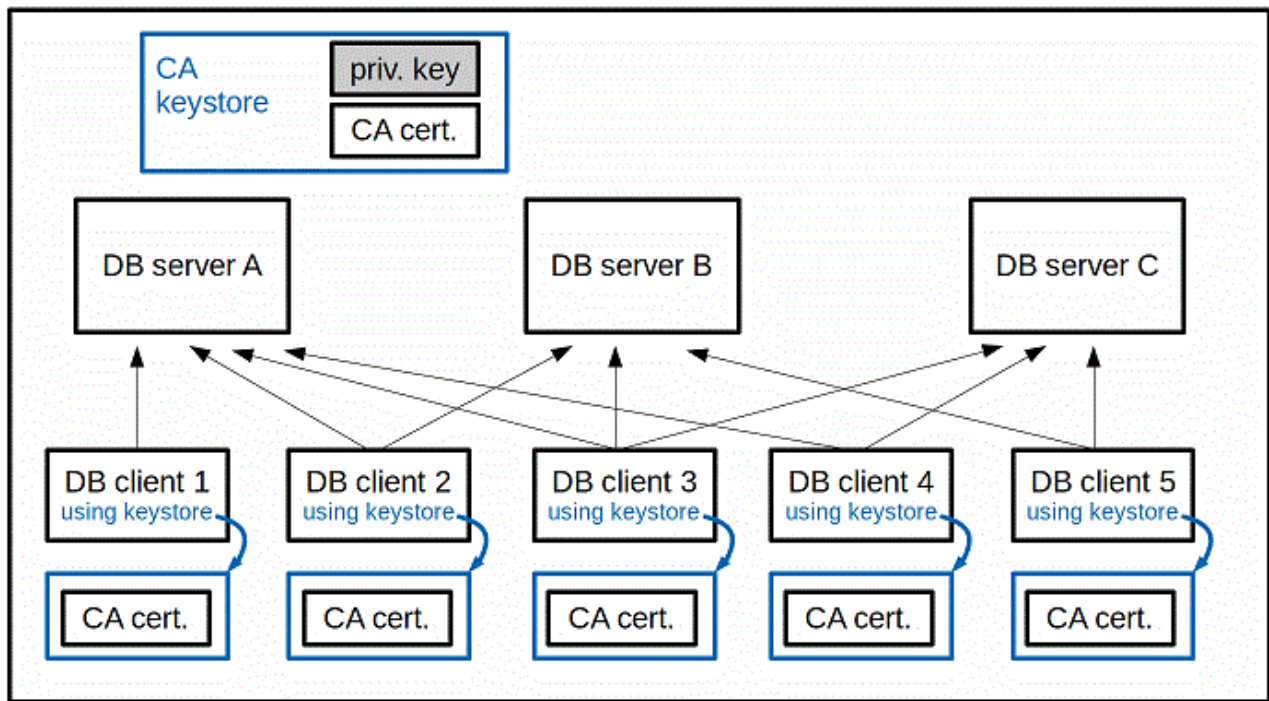
Setting up a home-grown CA is not very difficult. All that this CA needs is a private key and the corresponding certificate that contains the public key. In the scenario of a home-grown CA, there is little use for any intermediate CAs. Therefore, the certificate of the CA is the root CA certificate, which actually is a self-signed certificate. A self-signed root CA certificate is

created in the same way as a self-signed server certificate. For convenience (and security), the CA's private key and root CA certificate probably should be stored in its own keystore, just for the CA.

With the CA in place, the procedure for obtaining a user certificate for a database server pretty much is the same as it would be with a public CA. First, a private key needs to be generated for the server. Instead of using this private key to create a self-signed certificate, it is used to create a certificate request. This certificate request already contains most of the certificate content, including the public key that matches the server's private key. The certificate request is given to the CA. The CA uses the certificate request together with its own private key to create and sign the user certificate. The created user certificate is handed back to the database server. Now, the database server's keystore can be created with the server's private key and this user certificate.

In this case, the certificate is for a database server, it therefore also is called server certificate. But for the CA itself, there is no difference between a user certificate for a server or a user certificate for a client. They are just user certificates (as opposed to CA certificates). The CA may also hand its own CA certificate to the server, as any client will need this CA certificate to authenticate the server. In the home-grown CA environment, the administrator can directly distribute the CA certificate to the database clients. Or even better, create a keystore for the database clients that contains just this CA certificate and distribute this ready-to-use keystore to all the database clients.

Figure 6. Single CA certificate in client keystores for a 3-server environment



Compared to using self-signed server certificates the overhead of the CA's keystore has been added. But at the same time, the keystores of the clients have been simplified. In fact, all clients can just use a copy of the same keystore. However, the real advantage of using a (home-grown) CA shows in a dynamic database server landscape: Adding or removing database servers does not require any change in the keystores of the database clients. All that is needed for a new database server is

done on the server side: create a new private key, a new certificate request, have the CA create and sign the user certificate for the new database server using its own existing private key, then create the database server's keystore.

Examples for creating keystores using OpenSSL

This section demonstrates examples for creating keystores for TLS/SSL (Transport Layer Security / Secure Socket Layer) connections between database clients and servers. The examples in this topic use the tool and utilities provided by OpenSSL. The examples have been tested on Linux (x86 64-bit) as OS, using OpenSSL 1.1.1. For more information on OpenSSL, see <https://www.openssl.org/>.

Using the OpenSSL tool and utilities

Some general considerations regarding OpenSSL:

- openssl:

OpenSSL provides a single tool, "openssl", that is used at the command line to run different utilities for the various objects. Examples for such utilities used in this topic are:

- `genrsa`: generate a RSA private key
- `req`: generate certificate requests or self-signed certificates
- `x509`: handle or create and sign certificates of the X.509 standard.
- `pkcs12`: create or parse PKCS #12 keystore files
- `rand`: generate a sequence of random bytes
- `version`: display OpenSSL version information

- PEM

Privacy-Enhanced Mail (PEM) is a de facto file format for handling and storing cryptographic objects like private keys, certificates and certificate requests. It is a convenient format for use with "openssl" and its utilities, and therefore also used extensively in the examples. Cryptographic objects generally are ASN.1 and DER encoded objects, i.e. consist of binary data that includes non-printable character/byte sequences. PEM uses base64 encoding for binary data and adds one line for headers and footers to individual objects. The resulting text files are not really human readable, but can be handled easily, e.g. sending them in an e-mail.

- Random number generator

It may be necessary to create a file ".rnd" in the user's home directory, so that "openssl" can use its content as seed value for the random number generator. If the file is absent, an "openssl" command may fail with an error like the following:

```
Can't load ~/.rnd into RNG
140102054789568:error:2406F079:random number generator:RAND_load_file:
Cannot open file:./crypto/rand/randfile.c:88:Filename=~/.rnd
```

In this case, the file can easily be created by putting some random bytes into it, e.g. with the following command:

```
$ openssl rand 256 > ~/.rnd
```

The command generates a sequence of 256 random bytes and writes them to the file ".rnd" in the user's home directory. Afterwards, any "openssl" command that previously failed with the above error message can be repeated and should then run without producing the error.

- OpenSSL configuration

OpenSSL has a lot of configuration options. Many of them are setting default values that affect "openssl" commands by replacing absent command line options. Therefore, if an "openssl" command does not produce the desired result, it may be possible to specify an additional command line option rather than changing the system wide configuration for OpenSSL. Configuration changes may not only affect "openssl" commands, but also the functionality of the crypto library and its API functions. Changing of the OpenSSL configuration therefore should be done with utmost care, and in any case not without discussing things with the responsible system administrator.

With some "openssl" commands, it is also possible to specify an alternative configuration file on the command line. This can be a locally modified copy of the generic configuration file, just to use it with a particular command run. The name of the system wide configuration file is "openssl.cnf". The location of the file depends on the individual installation of OpenSSL. It can be determined with the following command:

```
$ openssl version -d
```

The file "openssl.cnf" can be found in the directory shown as OPENSSLDIR.

Setting up a CA with OpenSSL

To setup a CA with OpenSSL:

1. Create a RSA private key in a PEM file:

```
$ openssl genrsa -out rootCA1.key.pem -aes256 \
-passout pass:ca1passwd
```

The command creates the private key with the default size of 2048 bits and writes it encrypted in PEM format to the file "rootCA1.key.pem". The option "-aes256" tells the command to encrypt the private key using AES256 encryption. The encryption is based on the given password "ca1passwd".

The output PEM file can be parsed using the following command. As the private key is encrypted, the password for the content in the input file must be provided:

```
$ openssl rsa -in rootCA1.key.pem -passin pass:ca1passwd -text -noout
RSA Private-Key: (2048 bit, 2 primes)
modulus: ...
publicExponent: 65537 (0x10001)
privateExponent: ...
prime1: ...
prime2: ...
exponent1: ...
exponent2: ...
coefficient: ...
$
```

where the "..." stand for blocks of binary data in hex format that is undecipherable without much deeper knowledge of the algorithm internals for RSA keys.

2. Using the CA's private key, create a self-signed root CA certificate in a PEM file:

```
$ mydn="/C=US/ST=Florida/L=Anytown/O=Acme Software Inc./OU=Database CA/"
$ mydn=${mydn}"CN=Database CA Root1/emailAddress=dba_ca1@acme.info"
$ openssl req -new -x509 -key rootCA1.key.pem -passin pass:ca1passwd \
> -subj "${mydn}" -days 365 -out rootCA1.cert.pem
```

The command "openssl req" uses the previously generated private key to create a self-signed certificate and writes it to the output file "rootCA1.cert.pem". Because the private key in the key input file is encrypted, the password for the key input file must be provided. The validity of the certificate is one year beginning with the creation. The option "-x509" causes the command to create a self-signed certificate rather than a certificate request.

The option "-subj ..." specifies the distinguished subject name of the certificate owner. As this is a self-signed certificate, the issuer name automatically is the same as the subject name. For convenience, the longish subject name is given as the variable "\${mydn}", defined by the previous two shell commands. A distinguished name consists of several fields like country, state, location, organization, organizational unit, common name, etc. The format is a '/' that starts each field, followed by the field specifier, '=', and the field value string. "/C=US" therefore means field "country" has the value "US". Note that the individual fields have a maximum length. E.g. the country field can only have two characters for the 2-letter country code.

The following command can be used to parse the certificate in the PEM file and output its content:

```
$ openssl x509 -in rootCA1.cert.pem -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: ...
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = US, ST = Florida, L = Anytown, O = Acme Software Inc.,
           OU = Database CA, CN = Database CA Root1,
           emailAddress = dba_ca1@acme.info
    Validity
      Not Before: Jun  2 08:50:33 2021 GMT
      Not After  : Jun  2 08:50:33 2022 GMT
    Subject: C = US, ST = Florida, L = Anytown, O = Acme Software Inc.,
            OU = Database CA, CN = Database CA Root1,
            emailAddress = dba_ca1@acme.info
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus: ...
      Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        10:6B:B1:E1:C9:D9:9A:7F:B4:FF:9C:16:77:DD:56:9E:A1:58:6C:01
      X509v3 Authority Key Identifier:
        keyid:
          10:6B:B1:E1:C9:D9:9A:7F:B4:FF:9C:16:77:DD:56:9E:A1:58:6C:01

      X509v3 Basic Constraints: critical
```

```

CA:TRUE
Signature Algorithm: sha256WithRSAEncryption
...
$

```

**Note:**

- As a root CA certificate, it is self-signed. Therefore, subject name and issuer name are the same. For the same reason, the X509 version 3 extensions Subject Key Identifier and Authority Key Identifier also are the same.
- The Signature Algorithm is "sha256WithRSAEncryption". A signature algorithm of lower quality, like "sha1WithRSAEncryption", may not be accepted by newer versions of crypto libraries. With "openssl", "sha256WithRSAEncryption" normally is chosen by default. If for some reason the "openssl" command created a certificate with a lower quality signature algorithm, it may be better to include the option "-sha256" in the creation command to ensure compatibility with different crypto libraries, especially for newer versions.
- The X509 version 3 extension Basic Constraints is present and has the attribute values critical and CA:TRUE. While the attribute critical is not so important, the attribute CA:TRUE is. Because this certificate is intended to be a (root) CA certificate, the extension attribute CA:TRUE is required by some crypto libraries, especially with newer versions. If the extension and attribute are missing, then the certificate later may not be considered a valid CA certificate. This is a typical cause for a client side authentication error during the TLS handshake. The error message can be something like "no CA certificate found", or similar.

The private key and corresponding certificate are all that is needed for a functioning CA. The objects are stored in separate PEM files, which may be a bit cumbersome for safekeeping, but the PEM format is most convenient for issuing and signing user certificates for database servers. The private key is encrypted in its PEM file and therefore sufficiently protected. It is possible to store both, the private key and the certificate, in a password protected PKCS #12 keystore, but the protection would not be better than in the encrypted PEM file. Instead, the key and certificate would need to be extracted again into PEM files in order to use them for issuing and signing user certificates.

Creating the keystore for a database server

About this task

As SSL server, a database server needs to own a private key and the corresponding certificate. Both items should be stored in a PKCS #12 keystore file. To avoid the disadvantages of using self-signed certificates (as explained in the topic "Concepts of Keys, Certificates and Keystores for TLS"), the database server should use certificates issued by a CA. However, requesting and receiving certificates from an established public CA may be a slow process and usually even costs real money. The compromise between these two possibilities is to use one's own CA - a CA that was set up as shown in the previous example.

1. Create a RSA private key for the database server in a PEM file:

```
$ openssl genrsa -out server1.key.pem
```

The command creates the private key with the default size of 2048 bits and writes it in PEM format to the output file "server1.key.pem". The PEM file can be parsed with a command like:

```
openssl rsa -in server1.key.pem -text -noout
```



Note: The private key in the PEM file is not encrypted.

2. Use the server's private key to create a certificate request:

```
$ mydn="/C=US/ST=Florida/L=Anytown/O=Acme Software Inc./OU=DB Servers/"
$ mydn=${mydn}"CN=DB Server 1/emailAddress=db1@acme.info"
$ openssl req -new -key server1.key.pem \
> -subj "${mydn}" -out server1.req.pem
```

The command "openssl req" uses the previously generated private key to create a new certificate request and writes it to the output file "server1.req.pem". The option "-subj ..." specifies the subject name of this database server. For convenience, the longish subject name is given as the variable "\${mydn}", defined by the previous two shell commands. This subject name is a distinguished name that consists of several fields like country, state, location, organization, organizational unit, common name, etc. The fields in the string are separated by a '/

The PEM file with the certificate request can be parsed with the following command:

```
$ openssl req -in server1.req.pem -text -noout
Certificate Request:
  Data:
    Version: 1 (0x0)
    Subject: C = US, ST = Florida, L = Anytown, O = Acme Software Inc.,
            OU = DB Servers, CN = DB Server 1,
            emailAddress = db1@acme.info
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus: ...
      Exponent: 65537 (0x10001)
    Attributes:
      a0:00
    Signature Algorithm: sha256WithRSAEncryption
    ...
$
```



Note: The Signature Algorithm is "sha256WithRSAEncryption". A signature algorithm of lower quality, like "sha1WithRSAEncryption", may not be accepted by newer versions of crypto libraries. With "openssl", "sha256WithRSAEncryption" normally is the default configuration. If for some reason the "openssl" command



created a certificate request with a lower quality signature algorithm, it may be a good idea to override the default configuration by adding the option "-sha256" to the command.

The PEM file with this certificate request now must be passed to the CA, so that the CA, based on the request, can issue and sign the user certificate.

3. As CA, use the certificate request to issue and sign the user certificate for the database server:

```
$ openssl x509 -req -inform PEM -in server1.req.pem -set_serial 5 \
> -CA rootCA1.cert.pem -CAkey rootCA1.key.pem -passin pass:ca1passwd \
> -days 365 -outform PEM -out server1.cert.pem
```

The command reads the input certificate request file "server1.req.pem" (as specified in PEM format) and writes the signed certificate to the output file "server1.cert.pem" (also in the specified PEM format). The certificate is valid for 1 year beginning with the creation time.

This command must be performed as the CA, because it needs the CA's root CA certificate as well as the CA's private key. Both are read from PEM files as specified by the options "-CA rootCA1.cert.pem -CAkey rootCA1.key.pem". Because the CA's private key in file "rootCA1.key.pem" is encrypted, it is also necessary that the CA running this command provides the password for the input PEM file with the CA's private key. When creating the CA's private key, the password specified for this PEM file was "ca1passwd".



Note: For compatibility reasons with different versions of different crypto libraries, it may be necessary to add the option "-extensions usr_cert" to the command. This option refers to section "[usr_cert]" in the OpenSSL configuration file. There, it is stated:

```
# These extensions are added when 'ca' signs a request. # This goes against PKIX guidelines but some CAs
do it and some software # requires this to avoid interpreting an end user certificate as a CA.
```

(With OpenSSL version 1.1.1, used to test the example commands, this option does not make a difference for the issued certificate.)

The CA then hands the newly issued user certificate as file "server1.cert.pem" back to the requester, i.e. to the database server. For completeness, the CA also hands over a copy of its own CA certificate in PEM format, i.e. file "rootCA1.cert.pem". (While with OpenSSL the database server does not need the CA certificate that signed its own user certificate, this may be required by some other security libraries. Therefore the CA generally should provide a copy together with the user certificate.)

4. Create the keystore for the database server:

The keystore for the database server must contain the database server's private key and the corresponding user certificate. The user certificate is the one just received from the CA. Both items are in PEM files, "server1.key.pem" and "server1.cert.pem", that can be readily used to create the keystore with the following command. The inclusion of the CA certificate in the keystore is optional for OpenSSL, but may be required by some other security library. Therefore the following command also uses file "rootCA1.cert.pem", so that the keystore is more compatible.

```
$ openssl pkcs12 -export -in server1.cert.pem -inkey server1.key.pem \
> -name server1 -certfile rootCA1.cert.pem -caname rootCA1 \
> -passout pass:s1passwd -out server1.p12
```

The option "-export" causes the command to export the content read from the three input files into a PKCS #12 format keystore. The output file is "server1.p12" and the keystore is protected with password "s1passwd". The option "-name server1" tells the command to use "server1" as value for the friendly name attribute of the Safebags with the database server's private key and the user certificate. The option "-caname rootCA1" provides the friendly name attribute value "rootCA1" for the SafeBag with the CA certificate.

The content of the new keystore in file "server1.p12" can be parsed and examined with the following command:

```
$ openssl pkcs12 -in server1.p12 -passin pass:s1passwd -nodes -info
MAC: sha1, Iteration 2048
MAC length: 20, salt length: 8
PKCS7 Encrypted data: pbeWithSHA1And40BitRC2-CBC, Iteration 2048
Certificate bag
Certificate bag
PKCS7 Data
Shrouded Keybag: pbeWithSHA1And3-KeyTripleDES-CBC, Iteration 2048
Bag Attributes
    friendlyName: server1
    localKeyID: B8 D8 27 ED 1C FF 82 5F 50 42 AA 5E 5D FF E8 A2 95 C4 17 A5
subject=C = US, ST = Florida, L = Anytown, O = Acme Software Inc.,
    OU = DB Servers, CN = DB Server 1, emailAddress = db1@acme.info
issuer=C = US, ST = Florida, L = Anytown, O = Acme Software Inc.,
    OU = Database CA, CN = Database CA Root1,
    emailAddress = dba_ca1@acme.info
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
Bag Attributes
    friendlyName: rootCA1
subject=C = US, ST = Florida, L = Anytown, O = Acme Software Inc.,
    OU = Database CA, CN = Database CA Root1,
    emailAddress = dba_ca1@acme.info
issuer=C = US, ST = Florida, L = Anytown, O = Acme Software Inc.,
    OU = Database CA, CN = Database CA Root1,
    emailAddress = dba_ca1@acme.info
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
Bag Attributes
    friendlyName: server1
    localKeyID: B8 D8 27 ED 1C FF 82 5F 50 42 AA 5E 5D FF E8 A2 95 C4 17 A5
Key Attributes:
-----BEGIN PRIVATE KEY-----
...
-----END PRIVATE KEY-----
```



Note:



- The output of the above command shows that there are two certificates and one private key in the keystore. The private key is encrypted with PBE (password based encryption) and stored in a "shrouded keybag". The friendlyName and the localKeyID attribute values of the SafeBags for the private key and the corresponding user certificate are matching. The SafeBag with CA certificate has different values for those attributes.
- It is possible to combine all three input PEM files into a single PEM file, e.g. with a simple "cat" command, and then for the "openssl pkcs12 -export" command only provide the name of the single file with the "-in ..." option, omitting the options "-inkey ..." and "-certfile ...". Still, the "-name ..." and "-caname ..." options are always needed, as otherwise the private key or one of the certificates may be stored without a friendlyName attribute and later not be accepted for use in the TLS handshake.
- For a private key, the corresponding user certificate must be contained in the input. Furthermore, only one private key with corresponding user certificate should be present. A keystore normally does not contain multiple pairs of private key and corresponding user certificate.
- If additional certificates are present, then they are considered CA certificates and also stored in the keystore. For each CA certificate in the input, an option "-caname ..." with an individual value must be provided. Multiple "-caname ..." option values are applied for the additional certificates in the order they appear on the command line.
- Each certificate can be stored only once in the keystore, with a distinct friendlyName attribute value. Duplicate certificates in the input therefore should be eliminated beforehand. Duplicate certificates in a keystore are considered an error.

5. Remove the PEM file with the database server's private key:

The database server's private key now is stored with password protection in the PKCS #12 keystore. But it is still unprotected in the PEM file "server1.key.pem". To prevent misuse of this private key, the PEM file should be removed, or otherwise specially protected. Alternatively, the private key PEM file can be created with password protection right away. See the step "Create a RSA private key in a PEM file:" in the previous example for "Setting up a CA with OpenSSL" where this was done for the CA's private key. If the private key PEM file is password protected, then the password must be supplied for every read access to the private key. I.e. option "-passin pass:..." would be needed for the commands when creating the certificate request and when creating the keystore.

Creating the keystore for a database client

About this task

The database client needs the CA certificate (chain) in order to authenticate the database server during the TLS handshake. Based on the examples above, the database server's user certificate was issued and signed by a home-grown CA. Therefore, the database client needs the CA certificate of this CA. As the CA used its root CA certificate to sign the database server's user certificate, there are no intermediate CAs involved. When authenticating the database server, the chain of certificates therefore is very simple. It consists only of the database server's user certificate and the CA's root CA certificate. The

database client receives the database server's user certificate during the TLS handshake, and therefore only needs the root CA certificate in its own keystore. With that, just the PEM file with the root CA certificate is needed to create the database client's keystore.

Create the keystore using the root CA certificate:

```
$ openssl pkcs12 -export -nokeys -in rootCA1.cert.pem \  
> -caname rootCA1 -passout pass:c1passwd -out client1.p12
```

The command reads the root CA certificate from the input file "rootCA1.cert.pem" and creates the keystore in output file "client1.p12". This keystore is protected by password "c1passwd". The option "-nokeys" tells the command that no private key is to be read (or expected) from the input file.

The option "-caname rootCA1" provides "rootCA1" as value for the friendly name attribute of the SafeBag container, that holds the certificate in the keystore. As there is only this single (root) CA certificate in the input file, only a single "-caname ..." option is needed in the command. Even though the keystore contains only a single certificate, it is necessary to specify the "-caname ..." option with a friendly name attribute value. Omitting the option results in a SafeBag without the friendly name attribute and the certificate being ignored during the TLS handshake. This then causes an authentication failure during the TLS handshake, because no suitable CA certificate was found in the keystore.

(As for the database client only CA certificate(s) are needed, the keystore neither contains a private key nor a user certificate that would correspond to a private key. It is therefore not necessary to create a private key and corresponding user certificate for the database client. This also implies that all certificates in the database client's keystore are CA certificates, where at least one of them should be a root CA certificate.)

The content of the new keystore in file "client1.p12" can be parsed and examined with the following command:

```
$ openssl pkcs12 -in client1.p12 -passin pass:c1passwd -nodes -info  
MAC: sha1, Iteration 2048  
MAC length: 20, salt length: 8  
PKCS7 Encrypted data: pbeWithSHA1And40BitRC2-CBC, Iteration 2048  
Certificate bag  
Bag Attributes  
    friendlyName: rootCA1  
subject=C = US, ST = Florida, L = Anytown, O = Acme Software Inc.,  
    OU = Database CA, CN = Database CA Root1,  
    emailAddress = dba_ca1@acme.info  
issuer=C = US, ST = Florida, L = Anytown, O = Acme Software Inc.,  
    OU = Database CA, CN = Database CA Root1,  
    emailAddress = dba_ca1@acme.info  
-----BEGIN CERTIFICATE-----  
...  
-----END CERTIFICATE-----
```

Now, the keystores for database server and database client have been created. The given examples are for a simple scenario where a home-grown CA is used to issue and sign the certificates for all database servers. For this, the home-grown CA uses a single self-signed root CA certificate. Therefore, database clients that have just this one root CA certificate in their keystore can connect to all database servers. The certificate chain used by the database client during the TLS handshake to authenticate the database server is trivial: it consists only of the database server's user certificate (received from the server during the TLS handshake) and the root CA certificate found in the database client's keystore.

Few details to know when using "openssl"

Modifying an existing keystore

With the command "openssl pkcs12" it is not really possible to modify an existing keystore, e.g. to add or remove a CA certificate or to renew an expired certificate. If something needs to be changed in a keystore, then the keystore must be re-created from scratch using "openssl pkcs12 -export ..." with input from PEM files. If the original PEM input files are no longer available, then it is possible to extract the content of the existing keystore as a whole or partially into PEM files.

Extracting objects from a keystore into PEM files

Extracting objects from an existing keystore into a PEM file can be useful when a keystore needs to be changed, but original PEM files are no longer available. The basic command "openssl pkcs12" parses a PKCS #12 input file and produces PEM output. It can therefore be used to extract keystore content into a PEM file for re-use with a new "openssl pkcs12 -export ..." command. As the PKCS #12 keystores are password protected, the correct password for the keystore must be supplied to access and parse the content of a keystore file.

Following are some examples for extracting objects from a keystore. The examples use the database server's keystore as it was created in the previous example. This keystore contains a private key and the corresponding user certificate as well as the (root) CA certificate, and the password for this keystore is "s1passwd".

- Extract only the user certificate from the keystore:

```
$ openssl pkcs12 -in server1.p12 -passin pass:s1passwd \  
> -clcerts -nokeys -out s1.ext1.pem
```

The command parses the PKCS #12 input file "server1.p12" using password "s1passwd" and writes the output in PEM format to file "s1.ext1.pem". The option "-clcerts" tells the command to output only the user certificate, but not the CA certificate. The option "-nokeys" causes the command to exclude the private key from the output.

The output file "s1.ext1.pem" contains the database server's user certificate in PEM format between the header and footer lines, like:

```
-----BEGIN CERTIFICATE-----  
MIIDrzCCApcCAQUwDQYJKoZIhvcNAQELBQAwaIx CzA JBgNVBAYTALVTMRAwDgYD  
...  
YLz7hya2c7No4sFBRsHSQXJPXFRo0hG8ThiJZesjehmUnfE=  
-----END CERTIFICATE-----
```

Besides the actual PEM object as shown above, the output file contains some information on the PEM object, like the attributes friendly name and localKeyID of the SafeBag containers in the PKCS #12 keystore. In case of a certificate, also the certificate's issuer name and subject name are included as human readable information. This makes it fairly easy to verify, that the desired object was extracted from the keystore. The additional information can be removed with a text editor, if this is deemed necessary. However, the header and footer lines marking the begin and end of the PEM content should remain intact. Usually, other "openssl" commands that accept a PEM file as input conveniently ignore the additional, human readable information.

The PEM output file can be further parsed by the appropriate "openssl" command. For certificates in a PEM file, this is the command "openssl x509", e.g.:

```
$ openssl x509 -in s1.ext1.pem -text -noout
Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number: 5 (0x5)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = US, ST = Florida, L = Anytown, O = Acme Software Inc.,
           OU = Database CA, CN = Database CA Root1,
           emailAddress = dba_cal@acme.info
    Validity
      Not Before: Jun 2 16:10:19 2021 GMT
      Not After : Jun 2 16:10:19 2022 GMT
    ...
```

Omitting the option "-nokeys" from the extract command includes the private key as separate PEM section in the output. Because by default private keys get protected with PBE (password based encryption), the command without "-nokeys" then prompts twice for the password to use for the private key. The interactive prompting for this password can be avoided by either providing the password on the command line as "-passout pass:..." or by not protecting the private key in the PEM file with the option "-nodes". PEM output files that contain unprotected private keys should not be left 'unattended'.

The PEM section with a PBE protected private key in an output file would look like:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFHDBOBgkqhkiG9w0BBQ0wQTApBgkqhkiG9w0BBQwwHAQIDRjU76XwLYUCAggA
...
1Dw3Imvms7AnpD+4i0GNAg==
-----END ENCRYPTED PRIVATE KEY-----
```

- Extract only the CA certificate(s) from the keystore:

```
$ openssl pkcs12 -in server1.p12 -passin pass:s1passwd \
> -cacerts -nokeys -out s1.ext2.pem
```

Here, the option "-cacerts" tells the command to only output CA certificates but no user certificate. Without the option "-nokeys", the private key would be included in the output as explained already above. Also as explained above, the certificate in the output PEM file "s1.ext2.pem" can be further parsed using the command "openssl x509 -in s1.ext2.pem -text -noout".

- Extract only the private key from the keystore:

```
$ openssl pkcs12 -in server1.p12 -passin pass:s1passwd -nocerts \
> -passout pass:pkextpw -out s1.ext3.pem
```

Here, the option "-nocerts" excludes all certificates from the output. With that, only the private key is included in the output. By default, the private key is PBE protected in the output and therefore, the password for the output is provided as "pkextpw" with the option "-passout pass:pkextpw". The PEM section in the output file "s1.ext3.pem" looks like the following:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFHDBOBgkqhkiG9w0BBQ0wQTApBgkqhkiG9w0BBQwwHAQINL3IaWjpZeACAggA
```

```
...
Jm0+Vpf0b/o6BNN9UfcHcA==
-----END ENCRYPTED PRIVATE KEY-----
```

Knowing that the database server's private key is a RSA key, the private key in the PEM output file can be parsed with the following command. As the private key in the PEM file is password protected, the password is needed:

```
$ openssl rsa -in s1.ext3.pem -passin pass:pkextpw -text -noout
RSA Private-Key: (2048 bit, 2 primes)
modulus:
 00:e1:7b:21:f4:dc:b1:be:d3:2a:5c:33:69:75:35:
...
 6b:ab
publicExponent: 65537 (0x10001)
privateExponent:
 07:2f:ee:98:28:84:bd:e9:6b:3b:3f:24:48:69:28:
...
 89
prime1:
 00:fb:5c:69:e2:c4:8e:b6:91:81:35:e6:bd:a2:71:
...
 52:73:58:17:9f:46:fd:d1:b7
prime2:
 00:e5:a4:72:1a:81:66:b8:b1:e0:6f:9a:4f:dc:ad:
...
 8d:ca:f8:d0:ab:34:cb:e5:ad
exponent1:
 70:e7:20:94:80:0f:4e:47:eb:0e:c7:f6:b3:c9:7a:
...
 d5:ea:62:a9:e3:5c:4c:df
exponent2:
 51:71:98:12:34:70:84:f2:79:01:bb:bd:b5:2b:68:
...
 b9:5f:b3:49:f5:08:97:65
coefficient:
 25:68:e6:7d:50:f3:40:02:69:03:44:bf:10:39:e3:
...
 71:2c:27:ed:c3:40:87:5e
```

It is possible to extract the private key without PBE protection by using the option "-nodes". In this case, no output password with option "-passout pass:..." is needed. The PEM section of an unprotected private key in the output file would look like:

```
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCbKYwggSiAgEAAoIBAQDheyH03LG+0ypc
...
usCJO_rhtwwVxLCftw0CHXg==
-----END PRIVATE KEY-----
```



Warning: PEM files containing unprotected private keys should not be left 'unattended'.

- Extracting all objects from a keystore into a single PEM file:

```
$ openssl pkcs12 -in server1.p12 -passin pass:s1passwd \
> -out s1.ext4.pem -passout pass:s1extpw
```


Using the input password "s1passwd", the command parses the PKCS #12 input file "server1.p12" and writes all objects found in the keystore to the PEM output file "s1.ext4.pem". The private key in the PEM output file is PBE protected with the output password "s1extpw". As expected, the output file contains the encrypted private key, the user certificate that corresponds to the private key, and the root CA certificate.

As with the previous example, using option "-nodes" instead of option "-passout pass:s1extpw" would write the private key without PBE protection to the output file.

Using a single PEM input file to create a keystore

The previous example to create the database server's keystore used three PEM input files, one with the database server's private key, one with the database server's user certificate that corresponds to the private key, and one PEM file with the root CA certificate. An alternative way is to first combine all three input files into a single PEM file and then use it as single input file to create the keystore.

The output file from the extract command in the example above in fact is a single PEM file that contains everything for the database server's keystore. Therefore, it can be used to re-create the database server's keystore, in a different output file, with the following command:

```
$ openssl pkcs12 -export -in s1.ext4.pem -passin pass:s1extpw \  
> -name server1 -caname rootCA1 \  
> -out server1.extracted.p12 -passout pass:s1passwd
```

The command reads the PEM input file "s1.ext4.pem" that was created with the extract command in the previous example. Because in this PEM file the private key is PBE protected, the password "s1extpw" must be given with the option "-passin pass:...". The output is written to the new PKCS #12 keystore file "server1.extracted.p12" and for this keystore the password "s1passwd" is specified with the option "-passout pass:...". The options "-name server1" and "-caname rootCA1" provide the values for the friendly name attributes of the SafeBag containers in the keystore. The name "server1" is used for the two SafeBags containing the private key and the corresponding user certificate. The name "rootCA1" is used for the SafeBag holding the CA certificate.



Note: The options "-name ..." and "-caname ..." must be provided. Without these options, the SafeBags in the keystore would not have a friendly name attribute and hence the keystore would not work as expected during the TLS handshake. It could be argued, that the command could use the human-readable information in the PEM input file, as this also contains the friendly name attributes. However, this human-readable information is not really part of the PEM content. The "openssl pkcs12" command ignores such additional information and uses only the PEM content of the input file.

It becomes obvious, that the command also works with an input file that just contains the PEM content, including the PEM header and footer lines (i.e. without the human-readable information that was added by the extract command). Therefore, it is also possible to simply concatenate several PEM files with a private key, corresponding user certificates and several CA certificates into a single file, e.g. using the "cat" command. Such a combined PEM file can be used as single input file for the above command.

Extracting certificates for the database client from the database server's keystore

When using a CA to issue and sign the user certificate for a database server, then the database client needs the CA certificate (or the chain of CA certificates if intermediate CAs are involved) to authenticate the database server during the TLS handshake. If the CA certificate(s) are already in the database server's keystore (and the PEM files of the certificate(s) perhaps not at hand), it may be the easiest to extract the CA certificates from the database server's keystore and then use the extracted certificates(s) to create the database client's keystore. For the extraction, a command like the following can be used:

```
$ openssl pkcs12 -in server1.p12 -passin pass:s1passwd \  
> -cacerts -nokeys -out server1.cacerts.pem
```

However, in an environment where the database server uses a self-signed certificate and no CA is involved, things are slightly different. In this case, the database client needs the self-signed certificate of the database server to use it like a CA certificate during the TLS handshake. But from the viewpoint of the database server, this self-signed certificate is not a CA certificate, instead it is the database server's own user certificate. Therefore, to extract the self-signed certificate from the database server's keystore, the option "-clcerts" should be used instead of the option "-cacerts". The command therefore would look like the following:

```
$ openssl pkcs12 -in server2.p12 -passin pass:s2passwd \  
> -clcerts -nokeys -out server2.selfsignedcert.pem
```

Assuming a keystore for database server 2 that contains the database server's private key and the corresponding self-signed certificate, the command extracts this self-signed certificate and stores it in the PEM output file "server2.selfsignedcert.pem".

When creating the client keystore, the command "openssl pkcs12 -export ..." in both cases, for a self-signed certificate as well as CA certificate(s), needs the appropriate "-caname ..." option. In the database client's perspective, both certificates are CA certificates and are to be loaded without a corresponding private key.

X.509v3 certificate extension "Basic Constraints"

A certificate can contain several different extensions, so called "x509v3 extensions". One of them is the "Basic Constraints" extension that, depending on the version of a security library, can play a role in the TLS handshake. While older versions may not require that this extension is present in certificates, newer versions often do. The "Basic Constraints" extension can have two attributes: the attribute "critical" may be present or absent, the attribute "CA" can have two values, "true" or "false". While the "critical" attribute does not yet seem to be a required attribute with commonly used versions of security libraries, the attribute "CA" is required by some. Notably, current versions of IBM's GSKit as well as OpenSSL 1.1 require the "CA" attribute to be "true" for CA certificates. This means, a certificate that does not have the extension "Basic Constraints: CA:TRUE" is not considered a CA certificate and therefore ignored in a keystore.

It is good practice to make sure that all CA certificates as well as self-signed user certificates of database servers contain this extension "Basic Constraints: CA:TRUE". When using "openssl" to create certificates or certificate requests, the extension normally is inserted correctly by default. However, if you notice that your certificate creation or signing commands produce a CA certificate or self-signed certificate without the correct extension, then it may be worth the effort to figure out, why this happens and how to correct it.

The first check probably is in the OpenSSL configuration file, "openssl.cnf", located in the directory shown by command "openssl version -d". This file normally has different sections marked with "[section_label]". It is normal to have a configuration parameter "basicConstraints=CA:FALSE" in section "[usr_cert]", as this section is used when creating user certificates. Also section "[v3_req]" usually contains the parameter "basicConstraints=CA:FALSE". This section is used when creating certificate requests, and it is assumed that certificate requests are generally created for user certificates. On the other hand, section "[v3_ca]" is the section used by CAs and therefore has the parameter "basicConstraints = critical,CA:true". In addition, section "[req]" normally contains a parameter "x509_extensions = v3_ca" which tells the "openssl req" command to use section "[v3_ca]" also when creating self-signed certificates and therefore self-signed certificates normally get the correct extension.

If the default OpenSSL configuration file does not have the correct configuration parameters, or some other change is deemed necessary, then the default configuration file can be copied to a local file and modified. Then this locally modified file can be used at least to test changes. A local configuration file can be specified with option "-config ..." for "openssl req" commands, and with option "-extfile ..." for "openssl x509 -req" commands when signing certificates. Both commands also accept an option "-extensions ..." that can be used to specify a specific configuration file section to be used for the command.

Examples for creating keystores using IBM® GSKit

This topic demonstrates examples for creating keystores for TLS (Transport Layer Security) connections between database clients and servers. The examples in this topic use the tool "gsk8capicmd" provided by IBM's GSKit (Global Security Kit). This IBM® Global Security Kit (GSKit) provides libraries and utilities for TLS communication. GSKit version 8 is installed with the Informix® database server and database client products. The examples have been tested on Linux (x86 64-bit) as OS, using GSKit 8.0.55.26. For more information on GSKit as well as a complete reference for the "gsk8capicmd" tool please see the the "GSKCapiCmd User's Guide" at ftp://ftp.software.ibm.com/software/webserver/appserv/library/v80/GSK_CapiCmd_UserGuide.pdf.

IBM® GSKit is available for 32-bit as well as 64-bit architectures, and depending on the installed GSKit package, the "gsk8capicmd" utility has different names: simply "gsk8capicmd" for a 32-bit installation, and "gsk8capicmd_64" for a 64-bit installation. For simplicity, throughout this topic the utility is referred to just as "gsk8capicmd".



Remember: This topic is not a reference for the shown "gsk8capicmd" commands. Actual command syntax and functionality may be subject to change, especially with newer versions of GSKit.

The topic "Concepts of Keys, Certificates and Keystores for TLS" explains the advantages of using an in-house certificate authority (CA) vs. using only self-signed server certificates or involving a commercial CA. Therefore, this topic shows a simple example for setting up such an in-house CA.

Few things to know when using GSKit and gsk8capicmd

- [Certificate chain in the TLS server's keystore on page 68](#)
- [Dealing with commercial CAs on page 69](#)

- [Handling PEM files on page 69](#)
- [Using the command "gsk8capicmd -cert -import" needs extra caution on page 72](#)

Certificate chain in the TLS server's keystore

When using the GSKit libraries for TLS communication, the keystore of the TLS server must contain the complete certificate chain for the server's user certificate. This complete certificate chain is also used to validate the user certificate of the server. Such a validation can be performed manually. In the following example to illustrate this, a TLS server's keystore has a certificate chain with two intermediate CA certificates. The keystore contains:

```
$ gsk8capicmd -cert -list -db server2.p12 -pw s2passwd
Certificates found
* default, - personal, ! trusted, # secret key
! interB2
! interA2
! rootCA2
- server2
```

In the keystore, "rootCA2" is the root CA certificate that was used to sign and issue the intermediate CA certificate "interA2". This intermediate CA certificate "interA2" in turn was used to sign and issue the intermediate CA certificate "interB2". Finally, the intermediate CA certificate "interB2" was used to sign and issue the TLS server's user certificate "server2". The certificate chain therefore is "rootCA2" -> "interA2" -> "interB2" -> "server2".

Validating the user certificate "server2" in this keystore is done with the command:

```
$ gsk8capicmd -cert -validate -db server2.p12 -pw s2passwd -label server2
OK
```

With a successful validation of this user certificate in the keystore, the TLS server can use this user certificate "server2" in a TLS communication. If the validation of the user certificate is not successful, e.g. because one of the CA certificates in the chain is missing from the keystore, the user certificate would not function properly for the TLS server's communication. For instance, if the intermediate CA certificate "interA2" would be missing in the keystore, then neither the user certificate "server2" nor the intermediate CA certificate "interB2" could be validated successfully.

When adding certificates to a keystore, it is best to do this in such an order that the certificates always can be validated successfully. For the keystore of this example, the certificates are put into the keystore with four subsequent commands, each one handling a single certificate in a separate input file:

1. Add root CA certificate "rootCA2"
2. Add intermediate CA certificate "interA2"
3. Add intermediate CA certificate "interB2"
4. Receive user certificate "server2"



Note: With "gsk8capicmd" a user certificate is not simply "added" to the keystore. The keystore already contains the corresponding private key for the new user certificate, created implicitly with the certificate request. When adding the new user certificate to the keystore, it must be properly associated with the private key. Also, the certificate



request in the keystore gets removed when the new user certificate is put into the keystore. All this is done correctly by "gsk8capicmd" with the "receive" operation. How to do this in detail is explained later with an example.

Dealing with commercial CAs

Even though the topic is about setting up and using an in-house CA rather than a commercial CA, there is not really much difference between the two cases. Even for an in-house CA it may be a different department that is responsible for it. Understanding how to deal with a CA in general, in-house or commercial, can be of great help.

A user certificate is signed and issued by a CA based upon a certificate request. With "gsk8capicmd", a certificate request, and with it implicitly also the corresponding private key, is created directly in the keystore. As the certificate request needs to be given to a CA, the certificate request also gets written as a copy to a PEM file. This PEM file can be given conveniently to the CA. The private key and the (original) certificate request remain in the keystore and with that in the sole possession of the requestor. The CA signs and issues a new user certificate based upon the received certificate request. The CA then sends this new certificate, commonly in a PEM file, to the requestor. As seen above, to use this new certificate in a keystore, also the CA certificates used for the signing of the new certificate are needed by the requestor. The CA therefore sends these CA certificates as well, usually also in PEM format. A CA can send each of the certificates in a separate PEM file, which makes it easy to add them correctly into the keystore as described above. However, a CA also may choose to send all the certificates in a single PEM file. And unfortunately, the order of certificate objects in a PEM file is of no significance. Therefore, it cannot be counted on that the certificates in a PEM file received from a CA are in the correct order that is needed to put them into the keystore.

When receiving certificates from a CA in a PEM file, it is therefore necessary to understand the content of the PEM file, i.e. which certificates are in the PEM file, and to identify each one of them.

Handling PEM files

PEM (Privacy-Enhanced Mail) is a de facto file format for handling and storing cryptographic objects like private keys, certificates and certificate requests. Cryptographic objects generally are ASN.1 and DER encoded objects, i.e. consist of binary data that includes non-printable character/byte sequences. PEM uses base64 encoding for binary data and adds one line for headers and footers to individual objects. The resulting text files are not really human readable, but can be handled easily, e.g. sending them via e-mail. The PEM file format is commonly used by commercial CAs to receive certificate requests and send certificates.

A PEM file can contain just a single object, like a private key or a certificate, or it can contain multiple objects. The base64 encoded binary data of each object in a PEM file is enclosed with "comment lines" that denote whether the enclosed object is a private key, an encrypted private key or a certificate. Following is a PEM file that contains one private key (not encrypted) and two certificates:

```
-----BEGIN PRIVATE KEY-----
MIIIEvgIBADANBgkqhkiG9w0BAQEFAASCBAgEAAoIBAQDtSfmahK7hu0FF
StVHIryLTNoFPvSbBv0gre0Ea3tjPh5wxrNIYmgbVDbUR+smwNivcnMrIW+YjQVa
...
6qWLSVZPhAXuubHHJanD5pkz10+Igea4ba0IK1iMRTzC2SC/00gZRdCq5JiT/3TF
2KGgo26SkHkn09GVipJrkJ6M
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
```

```

MIIETFCAv2gAwIBAgII fBXp0DHqf50wDQYJKoZIhvcNAQELBQAwwZ8xCzAJBgNV
BAYTAlVTMRAdDgYDVQQIEwdG9yaWRhMRAdDgYDVQQHEwdBbnl0b3duMRswGQYD
...
/wE5Ft38ENmy4gzFNgsF9F0C/Keo4aV8ZD+B2o+dUstqGa1da34rQtA+Eqs4K6Vr
5/h7MWeDk4i14N/6Iy1AxTr0aMilK4a07cX03LKNZj02PSiYcl1tmIM=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIID+DCCAUcGAWIBAgIBNTANBgkqhkiG9w0BAQsFADCBnzELMAkGA1UEBhMCVVMx
EDA0BgNVBAgTB0Zsb3JpZGExEDA0BgNVBACTB0FueXRvd24xGzAZBgNVBAoTEkFj
...
byJXygfgt0Fv0bWpt3K/f51XRkur90+jRFeZAK7fBWRsg4Jm8j0wxL4KcIcU7Axg
HeHliFDyTxs3JQYC
-----END CERTIFICATE-----

```

While from the comment lines it is obvious that there is one private key and two certificates, it is rather impossible to simply "read" with the naked eye more information about these objects. But often it is necessary to figure out, what the certificates are about, whether they both are CA certificates, or which one is a user certificate and whether such a user certificate is associated with the private key or not. It is possible that before the "-----BEGIN ...-----" comment line of each object there is some human readable extra information about the object, like a "localKeyID" or Subject and Issuer distinguished names of certificates. However, such additional information is optional. The above shown example of a PEM file is the bare minimum, but perfectly valid. Therefore, it is often necessary to be able to decipher the base64 encoded binary data in order to have more information about the objects in a PEM file.

The GSKit "gsk8capicmd" utility can decipher base64 encoded binary content in PEM files just for display, albeit in a somewhat rudimentary way. "gsk8capicmd" with options "-cert -details -file" can read and decode the binary data of a certificate in the specified PEM file and display the information in a readable manner. However, for a PEM file containing multiple objects, "gsk8capicmd" does this only for the first certificate found in the PEM file. Therefore, the first step is to split a PEM file with multiple objects into several files, each containing just a single object. Thanks to the base64 encoding of the binary data, this splitting can be accomplished with any text file editor. It is important to include the respective "-----BEGIN ...-----" and "----- END ...-----" comment lines in the splitted files. Splitting the above example PEM file in this way yields 3 files, for convenience named "key1.pem", "cert1.pem" and "cert2.pem". Now, each single PEM file with a certificate can be deciphered using "gsk8capicmd" as follows:

```

$ gsk8capicmd -cert -details -file cert1.pem
Label : CN=Database CA Root1/emailAddress=dba_cal@acme.info,OU=Database CA,O=Acme
Key Size : 2048
Version : X509 V3
Serial : 7c15e9d031ea7f9d
Issuer : "CN=Database CA Root1/emailAddress=dba_cal@acme.info,OU=Database CA,O=Acme"
Subject : "CN=Database CA Root1/emailAddress=dba_cal@acme.info,OU=Database CA,O=Acme"
Not Before : November 13, 2022 8:28:24 AM CST
Not After : November 14, 2023 8:28:24 AM CST
Public Key
30 82 01 22 30 0D 06 09 2A 86 48 86 F7 0D 01 01
...
D5 02 03 01 00 01
Public Key Type : RSA (1.2.840.113549.1.1.1)
Fingerprint : SHA1 :
89 A0 58 49 20 71 94 C0 9F 7C 4E 2B FE 7A E3 37
E8 30 50 24
Fingerprint : MD5 :
4F 06 3C E8 43 E6 5A 7F 80 34 C1 AA A8 89 62 DA

```

```

Fingerprint : SHA256 :
32 66 16 B2 F0 9E DB C9 AC 45 7D 01 D6 78 C2 62
BE 22 AA 00 5C 01 5C 98 84 E4 B9 E9 B5 0C D6 0B
Fingerprint : HPKP :
09J+0fhn2zKr3z40y7YEq+U072Wr/2PPHJIKkYeI060=
Extensions
basicConstraints
ca = true
critical
SubjectKeyIdentifier
keyIdentifier:
28 D0 B2 DD E3 F9 26 06 C9 56 76 5E 17 5E 0A 21
C4 9A 85 46
AuthorityKeyIdentifier
keyIdentifier:
28 D0 B2 DD E3 F9 26 06 C9 56 76 5E 17 5E 0A 21
C4 9A 85 46
authorityIdentifier:
authorityCertSerialNumber:
Signature Algorithm : SHA256WithRSASignature (1.2.840.113549.1.1.11)
Value
55 3F 85 B3 06 AE E8 C8 37 B7 09 35 D1 C4 15 9F
...
C5 CE DC B2 8D 66 3D 36 3D 28 98 72 5D 6D 98 83
Trust Status : Enabled

```

The distinguished name in the "Subject:" line states the owner of the certificate, the distinguished name in the "Issuer:" line states the issuer of the certificate, normally a CA. In the example shown, both distinguished names are equal, and it is clear that this is a self-signed certificate. Consequently, also the values of the "Extensions" "SubjectKeyIdentifier" and "AuthorityKeyIdentifier" are the same. The extension "basicConstraints" has the two attributes "critical" and "ca = true". Together with the "CA" in the distinguished name it can be assumed, that this is a root CA certificate. With that, the PEM file "cert1.pem" can now be given a more mnemonic file name, e.g. "rootCA1.cert.pem", as it contains the root CA certificate of "Database CA Root1" (seen in the common name field of the "Subject").

Looking at the second PEM file with a certificate, "cert2.pem", with the following command shows:

```

$ gsk8capicmd -cert -details -file cert2.pem
Label : OU=DB ServersCN\=DB Server 1/emailAddress\=db1@acme.info,O=Acme Software In
Key Size : 2048
Version : X509 V3
Serial : 35
Issuer : "CN=Database CA Root1/emailAddress\=dba_ca1@acme.info,OU=Database CA,O=Acme Software In
Subject : "OU=DB ServersCN\=DB Server 2/emailAddress\=db2@acme.info,O=Acme Software In
Not Before : November 15, 2022 5:02:07 AM CST
Not After : November 16, 2023 5:02:07 AM CST
Public Key
30 82 01 22 30 0D 06 09 2A 86 48 86 F7 0D 01 01
...
41 02 03 01 00 01
Public Key Type : RSA (1.2.840.113549.1.1.1)
Fingerprint : SHA1 :
50 52 0B F1 F4 35 5A C0 F5 1E 73 9C C8 AA 7E 03
A1 07 51 CB
Fingerprint : MD5 :
3C F4 33 8D 47 96 79 28 6D A0 92 39 6B 88 DA 16
Fingerprint : SHA256 :

```

```

82 41 66 29 DD 94 C4 22 C6 2B 1E C3 08 EF 9C 6E
B6 40 C9 C2 B1 BC 2F FA 8D FD C4 DB 14 90 D2 BE
Fingerprint : HPKP :
8PqRjTnsU3u6nvwNX/RGux9aI7TgJH0tkZYJtYly7vI=
Extensions
basicConstraints
ca = false
critical
AuthorityKeyIdentifier
keyIdentifier:
28 D0 B2 DD E3 F9 26 06 C9 56 76 5E 17 5E 0A 21
C4 9A 85 46
authorityIdentifier:
authorityCertSerialNumber:
SubjectKeyIdentifier
keyIdentifier:
F0 6A F8 9A 46 C0 76 5A 05 7D AD C4 5B 01 6A 44
1C CB F2 D7
Signature Algorithm : SHA256WithRSASignature (1.2.840.113549.1.1.11)
Value
A6 D1 5F A9 D4 86 A9 5B C0 90 30 B3 9D 57 40 86
...
14 EC 0C 60 1D E1 E5 88 50 F2 4F 1B 37 25 06 02
Trust Status : Enabled

```

Here, "Subject" and "Issuer" are different, as are "AuthorityKeyIdentifier" and "SubjectKeyIdentifier". Therefore, this is not a self-signed certificate. The extension "basicConstraints" has the attributes "critical" and "ca = false" which tell, that this is a user certificate. Furthermore, the "Issuer" and "AuthorityKeyIdentifier" match the "Subject" and "SubjectKeyIdentifier" of the first certificate shown before. This establishes, that this second certificate, i.e. the user certificate, was signed and issued with the first certificate, the root CA certificate. The PEM file "cert2.pem" can now be renamed to "server2.cert.pem" as it contains the user certificate of "DB Server 2" (seen in the common name field of the "Subject").

Though the example PEM file also contains a private key object, this is not of further interest for this topic of using GSKit. This is mainly because with GSKit private keys are created directly in a keystore and never leave the keystore. With "gsk8capicmd" there is no way to export a private key from a keystore, so there should not be a need to import a private key into a keystore. (It is possible to do all these things with OpenSSL. More details on this are described in the respective topic on using OpenSSL).

Using the command "gsk8capicmd -cert -import" needs extra caution

It is possible to import one or more certificates from a PEM file into a keystore using the options "-cert -import" of "gsk8capicmd". However, this command should be used cautiously. If the PEM input file contains multiple certificates, these certificates are imported to the keystore in the order in which they appear in the PEM file. Problem is, that only one option "-label ..." can be specified, providing a label name only for the first certificate in the PEM file. Subsequent certificates get imported with label names created from the respective subject distinguished name of each certificate. Such label names are rather unwieldy and can make identifying the certificates in the keystore difficult. (Labels for the objects in a keystore are a property of the keystore's internal structure. They are not an attribute in the objects (certificates or private keys) themselves. Therefore, a PEM file normally does not contain any label names that would make it easier to identify the objects. If a PEM file contains a label name (or friendly name) in the extra information of objects, then this was added by the utility that created the PEM file from a keystore.)

Also, the user certificate may be included in such a PEM file. If the import command specifies as label name the label of the certificate request in the keystore, but the corresponding user certificate is not the first certificate in the PEM file, then the certificate request and its corresponding private key in the keystore can get overwritten with the first certificate in the PEM file. If this happens, the private key is lost completely and the keystore is no longer usable, as the private key cannot be recovered. (The only help in this case would be restoring a copy of the keystore file from before the attempted import operation.)

If the private key of the keystore is lost, then a newly signed and issued user certificate for this private key also is useless. In this case, it is necessary to create a new certificate request (and with this implicitly a new private key), then use the new request to get a new user certificate from the CA. Basically, the complete procedure must be repeated from the beginning.

Rather than using the command "gsk8capicmd -cert -import" to import multiple certificates in a PEM file into the keystore, it is better to split such a PEM file into separate files each containing just one certificate as described above. And then adding or receiving each certificate with a separate "gsk8capicmd" and in the correct order.

Setting up an in-house CA with GSKit

About this task

The CA needs a private key and corresponding self-signed certificate for signing and issuing user certificates. These two items are created and kept in a PKCS #12 keystore which is created and owned by the CA itself.

Following three steps in this section are performed in the role of the CA:

1. Create an empty keystore for the CA:

```
$ gsk8capicmd -keydb -create -db rootCA1.p12 -pw r1passwd -type p12
```

The command creates an empty keystore in file "rootCA1.p12" with password "r1passwd". The "gsk8capicmd" utility can handle keystores of different formats. The option "-type p12" specifies the PKCS #12 format for the new keystore.

2. Create a root CA certificate and private key in the keystore:

```
$ mydn="C=US,ST=Florida,L=Anytown,O=Acme Software Inc.,OU=Database CA,"
$ mydn=${mydn}"CN=Database CA Root1/emailAddress=dba_ca1@acme.info"
$ gsk8capicmd -cert -create -db rootCA1.p12 -pw r1passwd -label rootCA1 \
> -dn "${mydn}" -size 2048 -ca true -sigalg SHA256WithRSA
```

As the certificate must be a root CA certificate, the command used creates a self-signed certificate and implicitly the corresponding private key in the keystore "rootCA1.p12". To access the keystore, the password "r1passwd" is provided. The label (aka "friendly name") of the new certificate is "rootCA1". This label name is later used to refer to this certificate for further operations.

The option "-dn ..." specifies the distinguished subject name of the certificate owner. As this is a self-signed certificate, the issuer name automatically is the same as the subject name. For convenience, the longish subject name is given as the variable "\${mydn}", defined by the previous two shell commands. A distinguished name consists of several fields like country, state, location, organization, organizational unit, common name, etc. The fields are composed of the field specifier, '=', and the field value string. "C=US" therefore means field "country" has the value

"US". The fields are separated by comma. Note that the individual fields have a maximum length. E.g. the country field can only have two characters for the 2-letter country code.

The size of the private key is given as 2048 bit. The option "-ca true" includes the X509 version 3 extension "basicConstraints" with the attribute values "critical" and "ca = true". The option "-sigalg SHA256WithRSA" specifies the signature algorithm "SHA256WithRSASignature" to be used for the certificate. These three options ensure that the certificate is compatible with different crypto libraries, especially for newer versions.

The following command can be used to list the content of the keystore:

```
$ gsk8capicmd -cert -list -db rootCA1.p12 -pw r1passwd
Certificates found
* default, - personal, ! trusted, # secret key
- rootCA1
```

The certificate with label "rootCA1" is listed as a "personal" certificate, because it is associated with its corresponding private key in the keystore.

Details on the certificate with label "rootCA1" can be obtained with this command:

```
$ gsk8capicmd -cert -details -db rootCA1.p12 -pw r1passwd -label rootCA1
Label : rootCA1
Key Size : 2048
Version : X509 V3
Serial : 7c15e9d031ea7f9d
Issuer : "CN=Database CA Root1/emailAddress=dba_cal@acme.info,OU=Database CA,O=Acme"
Subject : "CN=Database CA Root1/emailAddress=dba_cal@acme.info,OU=Database CA,O=Acme"
Not Before : November 13, 2022 8:28:24 AM CST
Not After : November 14, 2023 8:28:24 AM CST
Public Key
 30 82 01 22 30 0D 06 09 2A 86 48 86 F7 0D 01 01
...
Public Key Type : RSA (1.2.840.113549.1.1.1)
Fingerprint : SHA1 :
 89 A0 58 49 20 71 94 C0 9F 7C 4E 2B FE 7A E3 37
 E8 30 50 24
Fingerprint : MD5 :
 4F 06 3C E8 43 E6 5A 7F 80 34 C1 AA A8 89 62 DA
Fingerprint : SHA256 :
 32 66 16 B2 F0 9E DB C9 AC 45 7D 01 D6 78 C2 62
 BE 22 AA 00 5C 01 5C 98 84 E4 B9 E9 B5 0C D6 0B
Fingerprint : HPKP :
 09J+0fhn2zKr3z40y7YEq+U072Wr/2PPHJikKyeI060=
Extensions
  basicConstraints
    ca = true
    critical
  SubjectKeyIdentifier
    keyIdentifier:
 28 D0 B2 DD E3 F9 26 06 C9 56 76 5E 17 5E 0A 21
 C4 9A 85 46
  AuthorityKeyIdentifier
    keyIdentifier:
 28 D0 B2 DD E3 F9 26 06 C9 56 76 5E 17 5E 0A 21
 C4 9A 85 46
    authorityIdentifier:
```

```

authorityCertSerialNumber:
Signature Algorithm : SHA256WithRSASignature (1.2.840.113549.1.1.11)
Value
55 3F 85 B3 06 AE E8 C8 37 B7 09 35 D1 C4 15 9F

```

As a root CA certificate, it is self-signed. Therefore, subject name and issuer name are the same. For the same reason, the X509 version 3 extensions SubjectKeyIdentifier and AuthorityKeyIdentifier also are the same. The "Extension" named "basicConstraints" has two attributes, "ca = true" and "critical". The attribute "ca = true" makes sure that this certificate can be used to sign new certificates, as well as to validate them later. By default, the new certificate is valid for one year, beginning with the time of the certificate creation. A different validity period can be specified with the option "-expire

". However, the start time of the validity is always the time of the certificate creation. (In fact, the start time is one day earlier to avoid otherwise possible issues with different time zones.)

3. Extract the root CA certificate (without private key) into a PEM file:

```

$ gsk8capicmd -cert -extract -db rootCA1.p12 -pw r1passwd -label rootCA1 \
> -target rootCA1.cert.pem -format ascii

```

The command extracts the certificate with label name "rootCA1" from the keystore file "rootCA1.p12". To access this keystore, password "r1passwd" is provided. The certificate is written in PEM format, as specified with option "-format ascii" into the file "rootCA1.cert.pem". This PEM file is later used to add the root CA certificate to the keystores of the database server and database client. When signing user certificates with this "rootCA1" keystore, the "rootCA1" certificate extracted here then should be distributed to the users together with the newly issued user certificate.

Creating the keystore for a database server

About this task

As TLS server, a database server needs to own a private key and the corresponding certificate. Both items should be stored in a PKCS #12 keystore file. To avoid the disadvantages of using self-signed certificates (as explained in the topic "Concepts of Keys, Certificates and Keystores for TLS"), the database server should use certificates issued by a CA. However, requesting and receiving certificates from an established commercial CA may be a slow process and usually even costs real money. The compromise between these two possibilities is to use the in-house CA that was set up as shown above.

1. Create an empty keystore for the database server:

```

$ gsk8capicmd -keydb -create -db server1.p12 -pw s1passwd -type p12

```

The command creates the empty keystore in file "server1.p12". This keystore is protected with password "s1passwd". The option "-type p12" specifies the format PKCS #12 for the keystore.

2. Create the database server's certificate request and private key in the keystore:

```

$ mydn="C=US,ST=Florida,L=Anytown,O=Acme Software Inc.,OU=DB Servers"
$ mydn=${mydn}"CN=DB Server 1/emailAddress=db1@acme.info"
$ gsk8capicmd -certreq -create -db server1.p12 -pw s1passwd -label server1 \
> -dn "${mydn}" -size 2048 -file server1.req.pem -sigalg SHA256WithRSA

```

The command creates a new certificate request and implicitly the corresponding private key in the keystore file "server1.p12". The certificate request has the label name "server1" and specifies as signature algorithm "SHA256WithRSA". The size of the private key is 2048 bit. The option "-file ..." writes the certificate request also to the output file "server1.req.pem" in PEM format.

The option "-dn ..." specifies the subject name of this database server. For convenience, the longish subject name is given as the variable "\${mydn}", defined by the previous two shell commands. This subject name is a distinguished name that consists of several fields like country, state, location, organization, organizational unit, common name, etc. The fields in the string are separated by comma.

The content of the keystore can be shown with the following command:

```
$ gsk8capicmd -certreq -list -db server1.p12 -pw s1passwd
Certificates requests found
    server1
```

Details about the certificate request with label "server1" can be obtained with this command:

```
$ gsk8capicmd -certreq -details -db server1.p12 -pw s1passwd -label server1
Label : server1
Key Size : 2048
Subject : "OU=DB ServersCN=DB Server 1/emailAddress=db1@acme.info,O=Acme Software
Public Key
    30 82 01 22 30 0D 06 09 2A 86 48 86 F7 0D 01 01
    ...
Public Key Type : RSA (1.2.840.113549.1.1.1)
Fingerprint :
d2e35afe6abf8bdf8d9d00ea6bde33e9
c1030a7a
Attributes
Signature Algorithm : SHA256WithRSASignature (1.2.840.113549.1.1.11)
Value
    C1 53 33 3C C1 0F E6 26 25 D8 45 E5 C4 A7 AF 93
    ...
```

3. As CA sign the new certificate requested by the database server:



Note: This step is performed in the role of the CA.

```
$ gsk8capicmd -cert -sign -db rootCA1.p12 -pw r1passwd -label rootCA1 \
-target server1.cert.pem -format ascii -file server1.req.pem \
-ca false -sigalg SHA256WithRSA
```

The command uses the keystore owned by the CA, file "rootCA1.p12", with the corresponding password "r1passwd". The option "-label ..." specifies that the certificate and corresponding private key with label name "rootCA1" are to be used for the signing of the new certificate. The new certificate is written to the target file "server1.cert.pem", in PEM format as per the given option "-format ascii". The new certificate is created based on the certificate request from the database server, provided in file "server1.req.pem".

The option "-ca false" makes the new certificate a user certificate, in GSKit terminology also called a personal certificate. With that, the new certificate can be used by an TLS server, but cannot be used as an intermediate CA certificate.

The option "-sigalg SHA256WithRSA" ensures that the new certificate is signed with the algorithm "SHA256WithRSASignature" to make it compatible with different crypto libraries.

By default, the new certificate is valid for one year, beginning with the time of issuing the certificate. A different validity period can be specified with the option "-expire ". However, the start time of the validity is always the time of issuing the certificate. (In fact, the start time is one day earlier to avoid otherwise possible issues with different time zones.)

As the CA, the newly issued user certificate is handed together with the previously extracted "rootCA1" certificate in file "rootCA1.cert.pem" to the user who requested the new certificate.

4. Put the root CA certificate into the database server's keystore:

After having received the requested user certificate together with the root CA certificate from the CA, these certificates can now be put into the database server's keystore, beginning with the root CA certificate in file "rootCA1.cert.pem".

In order to make the database server's keystore functional for TLS communication with GSKit, this keystore not only needs to contain the database server's own user certificate (aka personal certificate), but also the complete chain of certificates that were used to sign the database server's user certificate. In the example at hand, the database server's user certificate was signed directly with the root CA certificate. No intermediate CA certificates were used. Therefore, it is sufficient to just put this root CA certificate into the database server's keystore. It is best to do this step first, i.e. before also adding the database server's user certificate to the keystore. This way, the chain of CA certificates (in the example just the root CA certificate) is already available in the keystore when adding the new user certificate. With that the user certificate can be validated immediately when adding it.

```
$ gsk8capicmd -cert -add -db server1.p12 -pw s1passwd -label rootCA1 \
-file rootCA1.cert.pem -format ascii
```

The command adds the root CA certificate to the database server's keystore in file "server1.p12" using password "s1passwd". The certificate is stored in the keystore with label name "rootCA1". The certificate to add is in the input file "rootCA1.cert.pem", a PEM file as per the option "-format ascii". This is the PEM file with the root CA certificate that was extracted from the CA's keystore as last step of setting up the in-house CA described above.

The label name "rootCA1" is by choice the same as the label name of the same root CA certificate in the CA's keystore. This allows for quickly identifying the root CA certificate in the database server's keystore. The two keystores (of the database server and the CA) are completely separate keystores. Therefore, the label name of the root CA certificate can be diverse in the two keystores. Choosing the same label name is done just for convenience.

The root CA certificate in the database server's keystore can now be seen with this command:

```
$ gsk8capicmd -cert -list -db server1.p12 -pw s1passwd
Certificates found
```

```
* default, - personal, ! trusted, # secret key
! rootCA1
```

As expected, the certificate is shown as "trusted", which means that it is a CA certificate. Details of the certificate can be shown with the command:

```
$ gsk8capicmd -cert -details -db server1.p12 -pw s1passwd -label rootCA1
```

The output is the same detail information shown for the same certificate in the CA's keystore (see above).

5. Add the database server's own user certificate to the database server's keystore:

The database server's keystore still contains the certificate request and the corresponding private key. The new certificate of the database server, signed and issued by the CA in step 3 above and received from the CA in file "server1.cert.pem", needs to be associated with the private key and the certificate request. Therefore, the new certificate is not merely "added" to the keystore, instead it is "received". The options "-cert -receive" accomplish this operation correctly:

```
$ gsk8capicmd -cert -receive -file server1.cert.pem -format ascii \
-db server1.p12 -pw s1passwd
```

The command "receives" the database server's user certificate in file "server1.cert.pem" of format PEM into the database server's keystore "server1.p12". Password "s1passwd" is used to access this keystore. This certificate in file "server1.cert.pem" was signed and issued by the CA in step 3 above.

The new user certificate in the keystore replaces the certificate request for which it was issued. After the receive operation, the new certificate is listed as personal certificate in the keystore:

```
$ gsk8capicmd -cert -list -db server1.p12 -pw s1passwd
Certificates found
* default, - personal, ! trusted, # secret key
! rootCA1
- server1
```

As can be seen, the new certificate is stored with the label name "server1" as this was the label name of the replaced certificate request. The certificate request is no longer in the keystore:

```
$ gsk8capicmd -certreq -list -db server1.p12 -pw s1passwd
No certificate requests were found
```

Details for the database server's certificate can be shown with the following command, specifying the label name "server1":

```
$ gsk8capicmd -cert -details -db server1.p12 -pw s1passwd -label server1
Label : server1
Key Size : 2048
Version : X509 V3
Serial : 35
Issuer : "CN=Database CA Root1/emailAddress=dba_ca1@acme.info,OU=Database CA,O=Acme
Subject : "OU=DB ServersCN=DB Server 1/emailAddress=db1@acme.info,O=Acme Software
Not Before : November 15, 2022 5:02:07 AM CST
Not After : November 16, 2023 5:02:07 AM CST
Public Key
 30 82 01 22 30 0D 06 09 2A 86 48 86 F7 0D 01 01
...
```

```

Public Key Type : RSA (1.2.840.113549.1.1.1)
Fingerprint : SHA1 :
    50 52 0B F1 F4 35 5A C0 F5 1E 73 9C C8 AA 7E 03
    A1 07 51 CB
Fingerprint : MD5 :
    3C F4 33 8D 47 96 79 28 6D A0 92 39 6B 88 DA 16
Fingerprint : SHA256 :
    82 41 66 29 DD 94 C4 22 C6 2B 1E C3 08 EF 9C 6E
    B6 40 C9 C2 B1 BC 2F FA 8D FD C4 DB 14 90 D2 BE
Fingerprint : HPKP :
    8PqRjTnsU3u6nvwNX/RGux9aI7TgJH0tkZYJtYly7vI=
Extensions
    basicConstraints
        ca = false
        critical
    AuthorityKeyIdentifier
        keyIdentifier:
    28 D0 B2 DD E3 F9 26 06 C9 56 76 5E 17 5E 0A 21
    C4 9A 85 46
        authorityIdentifier:
        authorityCertSerialNumber:
    SubjectKeyIdentifier
        keyIdentifier:
    F0 6A F8 9A 46 C0 76 5A 05 7D AD C4 5B 01 6A 44
    1C CB F2 D7
Signature Algorithm : SHA256WithRSASignature (1.2.840.113549.1.1.11)
Value
    A6 D1 5F A9 D4 86 A9 5B C0 90 30 B3 9D 57 40 86
    ...
Trust Status : Enabled

```

The size of the (private) key associated with the certificate is 2048 bit, as was requested when creating the certificate request. The subject distinguished name corresponds to the value specified for the request, as does the signature algorithm. The "basicConstraints" extension has the attributes "ca = false" and "critical". "ca = false" designates this certificate as a user certificate that cannot be used as a CA certificate to validate other certificates. The validity of the certificate is one year, which is the default when signing and issuing a certificate as CA.

Subject and issuer distinguished names differ, because this is not a self-signed certificate. Correspondingly, the "AuthorityKeyIdentifier" and "SubjectKeyIdentifier" also have different values.

Creating the keystore for a database client

About this task

The database client needs the CA certificate (chain) in order to authenticate the database server during the TLS handshake. Based on the examples above, the database server's user certificate was issued and signed by an `in\u0002house` CA. Therefore, the database client needs the CA certificate of this CA. As the CA used its root CA certificate to sign the database server's user certificate, there are no intermediate CAs involved. When authenticating the database server, the chain of certificates therefore is very simple. It consists only of the database server's user certificate and the CA's root CA certificate. The database client receives the database server's user certificate during the TLS handshake, and therefore only needs the root CA certificate in its own keystore. With that, just the PEM file with the root CA certificate is needed to create the database client's keystore.

1. Create an empty keystore for the database client:

```
$ gsk8capicmd -keydb -create -db client1.p12 -pw c1passwd -type p12
```

The command creates the empty keystore in file "client1.p12". This keystore is protected with password "c1passwd". The option "-type p12" specifies the format PKCS #12 for the keystore.

2. Add the root CA certificate to the database client's keystore:

```
$ gsk8capicmd -cert -add -db client1.p12 -pw c1passwd -label rootCA1 \
-file rootCA1.cert.pem -format ascii
```

The command adds the root CA certificate to the database client's keystore in file "client1.p12" using password "c1passwd". The certificate is stored in the keystore with label name "rootCA1". The certificate to add is in the input file "rootCA1.cert.pem", a PEM file as per the option "-format ascii". This is the PEM file with the root CA certificate that was extracted from the CA's keystore as last step of setting up the in-house CA described above.

The root CA certificate in the database client's keystore can now be seen with this command:

```
$ gsk8capicmd -cert -list -db client1.p12 -pw c1passwd
Certificates found
* default, - personal, ! trusted, # secret key
! rootCA1
```

The certificate is listed as "trusted". This means it is a proper CA certificate and can be used to validate certificates that were signed with it. When the database client receives the database server's user certificate during the TLS handshake, the root CA certificate in the database client's keystore is available to validate the received certificate and authenticate the database server.

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 *HCL® 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 *HCL® 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 *HCL® 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 *HCL® Informix® Administrator's Reference*.

Encrypting column data

You can store sensitive data in encrypted format.

About this task

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 on page 83](#).
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 84](#).

Results

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 84](#).

See the *HCL® 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';
```

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 the 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.

By default in root 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.

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 root 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 root installations.

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 *HCL® 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.

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.



Note: Support for Communication Support Module (CSM) is removed starting Informix Server 14.10.xC9 . You should use Transport Layer Security (TLS)/Secure Sockets Layer (SSL) instead.

If you want to support a single sign-on (SSO) environment, you can use the Generic Security Services CSM (GSSCSM) 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.

You can configure HCL® 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.

You can limit the ability of denial-of-service attacks to prevent legitimate connections to the database server from being blocked.

Related information

[Connections to a non-root installation \(UNIX, Linux\) on page 91](#)

[Configuring secure ports for connections between replication servers on page](#)

[Configuring secure connections for high-availability clusters on page](#)

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.

On UNIX™ and Linux™, the 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 HCL® 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.

The 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 information

[Internal users \(UNIX, Linux\) on page 87](#)

[REMOTE_SERVER_CFG configuration parameter on page](#)

[REMOTE_USERS_CFG configuration parameter on page](#)

[sqlhosts file and SQLHOSTS registry key options on page](#)

[Pluggable authentication modules \(UNIX or Linux\) on page 110](#)

[LDAP authentication support on Windows on page 113](#)

[Creating database server users \(UNIX, Linux\) on page 93](#)

[Maintaining database server users \(UNIX, Linux\) on page 94](#)

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.

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 information

[Authentication mechanisms on page 86](#)

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.

Users can be mapped DB-Access. 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 information

[USERMAPPING configuration parameter \(UNIX, Linux\) on page](#)

[Surrogate user properties \(UNIX, Linux\) on page](#)

[Pluggable authentication modules \(UNIX or Linux\) on page 110](#)

[Setting up an SSO authentication environment on page 123](#)

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 disallowing 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 information

[onmode -cache surrogates: Cache the allowed.surrogates file on page](#)

[CREATE USER statement \(UNIX, Linux\) on page](#)

[Surrogate user properties \(UNIX, Linux\) on page](#)

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.

Example

```
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.

Example

```
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.

Example

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 information

[onmode -cache surrogates: Cache the allowed.surrogates file on page](#)

[CREATE USER statement \(UNIX, Linux\) on page](#)

[Surrogate user properties \(UNIX, Linux\) on page](#)

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 information

[User mapping tables \(UNIX, Linux\) on page 96](#)

[ALTER USER statement \(UNIX, Linux\) on page](#)

[DROP USER statement \(UNIX, Linux\) on page](#)

[RENAME USER statement \(UNIX, Linux\) on page](#)

[SET USER PASSWORD statement \(UNIX, Linux\) on page](#)

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 information

[CREATE USER statement \(UNIX, Linux\) on page](#)

[ALTER USER statement \(UNIX, Linux\) on page](#)

[Surrogate user properties \(UNIX, Linux\) on page](#)

[USERMAPPING configuration parameter \(UNIX, Linux\) on page](#)

[Default name and location of the explain output file on UNIX on page](#)

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 *HCL® Informix® Installation Guide* *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 `efd` 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 `efd` 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 the 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 `efd` value in the `sqlhosts` file for the connection attempt to succeed.

Distributed queries on non-root installations

You can perform distributed queries on non-root server installations after you set the `REMOTE_USERS_CFG` or `REMOTE_SERVER_CFG` configuration parameter. The `REMOTE_USERS_CFG` configuration parameter specifies an alternative

to using `~/ .rhosts` files for listing trusted users on a remote server. `REMOTE_SERVER_CFG` configuration parameter specifies an alternative to using the `etc/hosts.equiv` file for listing trusted remote hosts.

Related information

[Connection security on page 84](#)

[Installation owner on page](#)

[REMOTE_SERVER_CFG configuration parameter on page](#)

[REMOTE_USERS_CFG configuration parameter on page](#)

[Creating database server users \(UNIX, Linux\) on page 93](#)

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.

Before you begin

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.

- You must have DBSA privileges. By default, the owner of a non-root 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.

About this task

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
```

What to do next

To enable 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`, `dbssso`, `ao` and `bagroup`.

Related information

[User mapping tables \(UNIX, Linux\) on page 96](#)

[Connections to a non-root installation \(UNIX, Linux\) on page 91](#)

[Maintaining database server users \(UNIX, Linux\) on page 94](#)

[Authentication mechanisms on page 86](#)

[USERMAPPING configuration parameter \(UNIX, Linux\) on page](#)

[CREATE USER statement \(UNIX, Linux\) on page](#)

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.

Before you begin

Prerequisites:

- You must have DBSA privileges.

About this task

To manage user accounts:

Run one of the following statements, as appropriate:

Choose from:

- 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.

Example

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 **jaya189**, by running this command:

```
SET USER PASSWORD OLD js12342394 NEW jaya189
```

Related information

[Creating database server users \(UNIX, Linux\) on page 93](#)

[Authentication mechanisms on page 86](#)

[User mapping tables \(UNIX, Linux\) on page 96](#)

[ALTER USER statement \(UNIX, Linux\) on page](#)

[DROP USER statement \(UNIX, Linux\) on page](#)

[RENAME USER statement \(UNIX, Linux\) on page](#)

[SET USER PASSWORD statement \(UNIX, Linux\) on page](#)

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 HCL® Informix® access and control level of discretionary access privileges.

sysusermap table

Database: SYSUSER

Table 5. Schema of the sysusermap table

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

sysurrogates table

Database: SYSUSER

Table 6. Schema of the sysurrogates table

Col	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 HCL® 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 <i>gid</i> 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.

Table 6. Schema of the syssurrogates table (continued)

Column	Type	Description
group name	CHAR(32)	A group name that exists on the operating system of the HCL® Informix® host computer.
home dir	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 7. 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.
group name	CHAR(32)	A group name that exists on the operating system of the HCL® Informix® host computer.
group seq	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 8. Schema of the sysintauthusers table

Column	Type	Description
username	NCHAR(32)	Name for the user.
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
account_expire	DATETIME YEAR TO SECOND	N/A

Related information

[Creating database server users \(UNIX, Linux\) on page 93](#)

[USERMAPPING configuration parameter \(UNIX, Linux\) on page](#)

[Maintaining database server users \(UNIX, Linux\) on page 94](#)

[Internally authenticated users \(UNIX, Linux\) on page 90](#)

Guest account (Windows™)

Disable the Windows™ Guest account to prevent anonymous logins.

By default, the Windows™ Guest account is disabled. If you enable the Guest account, remote and local users can connect to the database server anonymously. Disable the Guest account in the Windows™ Control Panel. Also, set the configuration parameter `SECURITY_LOCAL_CONNECTION = 1` to prevent local users from connecting by specifying a user name that is different from the user who is running the client process.

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 information

[CREATE TRUSTED CONTEXT statement on page](#)

[Database Security Administrator Role on page 138](#)

[Security Administration Options on page](#)

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 information

[CREATE TRUSTED CONTEXT statement on page](#)

[Database Security Administrator Role on page 138](#)

[TCP/IP connectivity files on page](#)

Creating a trusted-context object

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

Before you begin

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.

Choose from:

- 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 IBM® 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 the `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.

What to do next

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 information

[CREATE TRUSTED CONTEXT statement on page](#)

[ALTER TRUSTED CONTEXT statement on page](#)

[DROP TRUSTED CONTEXT statement on page](#)

[RENAME TRUSTED CONTEXT statement on page](#)

[Database Security Administrator Role on page 138](#)

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

Example

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 information

[CREATE TRUSTED CONTEXT statement on page](#)

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 information

[CREATE TRUSTED CONTEXT statement on page](#)

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.

Example

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 information[CREATE TRUSTED CONTEXT statement on page](#)[Roles on page](#)[User roles on page 132](#)[Default roles on page 133](#)

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.

Example

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 information[CREATE TRUSTED CONTEXT statement on page](#)[Roles on page](#)[User roles on page 132](#)

Creating a trusted connection

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

Before you begin

Before you can create a trusted connection, ensure that you have met the following prerequisites:

- 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.

Choose from:

- 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 information

[CONNECT statement on page](#)

[The SQLSetConnectAttr\(\) function with authentication on page](#)

Switching the user ID on a trusted connection

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

Before you begin

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:

Choose from:

- 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 information

[SET SESSION AUTHORIZATION statement on page](#)

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.

Example

Table 9. 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

Table 9. Rules for switching users on a trusted connection, and potential errors related to the rules. (continued)

Switching Rule	Related Errors
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 (UNIX™ or Linux™)

A Pluggable Authentication Module (PAM) is a well-defined framework for supporting different authentication modules that were 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 computer 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. You can stack many modules one after another to enable the application to be authenticated in multiple ways before the application grants 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) to avoid forcing major changes on users.

To use PAM with HCL® Informix®:

- Your Informix® database server must be on an operating system platform that supports PAM.
- If your client applications are written with Client SDK, the version of Client SDK must be sufficiently recent.
- If your client applications use Distributed Relational Database Architecture™ (DRDA®) connections, you can configure password authentication but not challenge-response authentication.
- 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.
- For Linux™ platforms, when you configure PAM to require both password and challenge-response authentication, the PAM service always ignores the password that is 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 computers or in separate locations on a single computer).

Related information

[Authentication mechanisms on page 86](#)

[Mapped users \(UNIX, Linux\) on page 87](#)

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 PAM

The pluggable authentication mode (PAM) determines whether a user can authenticate by providing a password, responding correctly to a challenge, or a combination of both.

You can use either authentication mode for Client SDK connections. You can use the password authentication mode for Distributed Relational Database Architecture™ (DRDA®) connections.

The PAM implementation in HCL® 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 it connects 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.

For Linux™ platforms, 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 connection to use PAM

You configure a connection to use a pluggable authentication module (PAM) by adding the PAM options to the connection information in the `sqlhosts` file.

About this task

To configure a connection to use PAM, you 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.

To configure PAM authentication for a connection:

Edit the `sqlhosts` file for the connection. Include the `s=4` option and the PAM options `pam_serv`, to specify the PAM service, and `pamauth`, to specify the authentication mode.

Example

The following `sqlhosts` file entry shows a connection that is 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 connection that is configured to use PAM password authentication:

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

The following `sqlhosts` file entry shows a Distributed Relational Database Architecture™ (DRDA®) connection that is configured to use PAM password authentication:

```
ifxserver3 drsoctcp serverdrda portnum3 s=4,pam_serv=(pam_pass),pamauth=(password)
```

Related information

[sqlhosts file and SQLHOSTS registry key options on page](#)

Define a PAM service

If you define your own PAM service for the Informix® server instead of using one of the services that are already defined in the `pam.conf` file, you must include the `auth` and `account` functions.

Include the `auth` and `account` functions for the service.

On UNIX™, add the following lines to the `pam.conf` file:

```
pam_service auth sufficient path_and_filename
pam_service account sufficient path_and_filename
```

where `pam_service` is the name of the PAM service and `path_and_filename` is the path and file name of your PAM file.

For example:


```
pam_service auth sufficient /opt/mycompany/lib/my pam.so
pam_service account sufficient /opt/mycompany/lib/my pam.so
```

See the operating system documentation for the complete PAM configuration instructions.

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 information

[Authentication mechanisms on page 86](#)

Installing and customizing the LDAP authentication support module

Before you begin

Before you can use the HCL® 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® Security Directory Server and openLDAP.

About this task

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 HCL® 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.

What to do next

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.

About this task

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 HCL Informix® for LDAP

About this task

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 HCL® 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 on page 115](#)
- [Application development for authentication modules on page 115](#)
- [Distributed transactions and authentication modules on page 117](#)
- [Client APIs and authentication support modules on page 118](#)
- [Compatibility issues with authentication modules on page 119](#)

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 HCL® 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_app1.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 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.

Informix provides two options to support distributed connection PAM authentication:

1. sysuser:sysauth

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 10. 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 server name, and one host name. Do not use comma-separated or space-separated lists of server or host names in one entry.

2. On Linux/Unix platforms through OS rhosts PAM module, for example, define your PAM configuration file as

```
auth sufficient pam_rhosts.so
auth required pam_unix.so
account required pam_unix.so
```

Client APIs and authentication support modules

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

The following HCL 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 HCL® 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

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

Compatibility issues with authentication modules

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

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

- HCL Informix®-4GL does not have a mechanism for identifying callback functions and therefore cannot directly support PAM or LDAP authentication. However, if HCL Informix®-4GL uses the correct version of CSDK, you can write C code that can be called from HCL 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 HCL® 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 `concsn.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 `concsn.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 *HCL® 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 *HCL® 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 126](#)

Configuring password encryption

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

HCL® 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

```
csmlib(" { client=clientlib ,server=serverlib | csmlib } " , " " , " [p= { 0 | 1 } ] ")
```

Option	Description
<code>clientlib</code>	The full path and name of the shared library that is the CSM on the client computer.

Option	Description
<i>csmlib</i>	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.
<i>csmname</i>	The name that you assign to the CSM.
p	<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.</p> <p>1</p> <p>The password is mandatory for authentication.</p> <p>0</p> <p>The password is not mandatory for authentication. If the client provides the password, it is encrypted and used for authentication.</p>
<i>serverlib</i>	The full path and name of the shared library that is the CSM on the database server.

SMI tables and conccsm.cfg setting

About this task

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 `conccsm.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 `conccsm.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 `conccsm.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 `conccsm.cfg` file.
5. Start the database server again with the `oninit` command.

Example conccsm.cfg entries for password encryption

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

The following two examples illustrate the two alternatives for parameters that you must enter in the `conccsm.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.

HCL® 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 the 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 HCL® 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.

Before you begin

About this task

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 124](#)).
3. Configure the `sqlhosts` information and Generic Security Services communications support module (GSSCSM) on the computer hosting the database server (see [Configuring the HCL Informix instance for SSO on page 125](#)).
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 on page 124](#)).

Example

What to do next

Related information

[Mapped users \(UNIX, Linux\) on page 87](#)

Clients supporting SSO

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

You can use the following clients with SSO:

- HCL Informix® ESQL/C
- HCL Informix® ODBC Driver
- HCL Informix® JDBC Driver
- HCL Informix® DB-Access

See [Configuring ESQL/C and ODBC drivers for SSO on page 129](#) and [Configuring JDBC Driver for SSO on page 130](#) 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.

Before you begin

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.

About this task



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 130](#) before you do the following steps.

You must have kadmin privileges (UNIX™ and Linux™) or domain administrator rights (Windows™) to complete steps [3 on page 125](#), [4 on page 125](#), and [5 on page 125](#).

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 the HCL Informix® instance for SSO

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

Before you begin

About this task

1. [Set SQLHOSTS information for SSO on page 125](#)
2. [Set up the `concsm.cfg` file for SSO on page 126](#)
3. [Ensure `keytab` file has the required key \(UNIX and Linux\) on page 127](#)
4. [Verify Informix uses Kerberos authentication for SSO on page 128](#)

Example

What to do next

Set SQLHOSTS information for SSO

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

Before you begin

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

About 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 *HCL® 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.

Example

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`:

```
o1_home2data onsoctcp jkent-005 csm=(GSSCSM)
```

Example

What to do next

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 conccsm.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.

HCL® 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 `conccsm.cfg` file.

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

Syntax

```
(explicit id unique_121_Connect_42_encp001) unique_121_Connect_42_encp001 csmname(" { client=clientlib ,server=serverlib | csmlib } " ,
"" , " [{"c"={ 0 | 1}] [{"i"={ 0 | 1}]}] )
```

Option	Description
<code>csmname</code>	The name that you assign to the CSM
<code>csmlib</code>	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
<code>clientlib</code>	The path and name of the shared library that is the CSM on the client computer

Option	Description
<i>serverlib</i>	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 information

[CSM configuration file on page 119](#)

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 begin

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.

About this task



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®.

Example

What to do next**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.

Before you begin

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.

About this task

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).

Example

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.

Example

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.

Example

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
o1_home2data/jkent-005.payroll.jkenterprises.com@jkenterprises.com

```

Configuring ESQ/C and ODBC drivers for SSO

The steps for preparing the SQLHOSTS information and the Generic Security Services (GSS) CSM configuration file for ESQ/C and ODBC and a client computer are similar to the corresponding server-side setup procedures.

Before you begin

Complete the tasks outlined in [Configuring the HCL Informix instance for SSO on page 125](#) before working on your client.

About this task

See [Clients supporting SSO on page 124](#) for a list of clients that support SSO with HCL® Informix®.

1. Complete any setup steps specific to the client software you are using. This includes the following steps:
 - a. For ESQ/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.

Example

For example, the following entry can be valid for the company JK Enterprises if the port number is already set in `$INFORMIXDIR/etc/services`:

```
o1_sso_krb  onsoctcp  jkent-005  o1_sso_svce  csm=(GSSCSM)
```

- b. 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:

Example

```

Driver=/usr/informix/lib/cli/iclit09b.so
Description=IBM INFORMIX ODBC DRIVER
Database=stores_demo
ServerName=o1_sso_krb
Options="csm=(GSSCSM)"

```

- c. *for ODBC on Windows™:*

Result

- i. Open SETNET32 (Start > All Programs > HCL Informix® Client-SDK > Setnet32) and select the Server Information tab.
- ii. 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.

- iii. Open the ODBC Driver manager program.
 - iv. 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 119](#) 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 126](#) 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.

1. Set the `DriverManager.getConnection()` method with the SSO options.

Example

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).

2. Create a login configuration file before you run 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.

Example

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
```

Related information

[The DriverManager.getConnection\(\) method on page](#)

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 11. SECURITY_LOCALCONNECTION configuration parameter settings

Setting	Explanation
0	No security checking occurs.
1	HCL® Informix® compares the user ID of the owner trying to connect with the connection user ID. If these do not match, HCL® Informix® rejects the connection.
2	HCL® Informix® performs the same checking that is performed when SECURITY_LOCALCONNECTION is set to 1. In addition, HCL® 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, HCL® Informix® establishes a connection only if the connection meets the requirements of the security check.

Limiting denial-of-service flood attacks

HCL® 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 *HCL® Informix® Administrator's Reference*.

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 on page 132](#).
- Control who can create databases. See [Setting permission to create databases on page 134](#).
- Prevent unauthorized users from registering user-defined routines. See [Security for external routines \(UDRs\) on page 134](#).
- Control whether other users besides the DBSA can view executing SQL statements. See [Enabling non-DBSA users to view SQL statements in an active session on page 135](#).

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 *HCL® Informix® Guide to SQL: Syntax*.

Related reference

[Example of assigning a default role in a trusted-context object on page 104](#)

[Example of assigning user-specific privileges in a trusted-context object on page 105](#)

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 *HCL® Informix® Guide to SQL: Reference*. For more information about role separation, see [Using role separation on page 187](#).

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 104](#)

Granting privileges for a default role

About this task

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`.

Results

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 *HCL® 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.

About this task

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:

Example

```
DBCREATE_PERMISSION informix
```

2. You can include multiple instances of **DBCREATE_PERMISSION** in the `onconfig` file to give additional users permission to create databases.

Example

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 *HCL® Informix® Guide to SQL: Syntax*.

Enabling non-DBSA users to view SQL statements in an active session

You can enable non-database server administrator (DBSA) users to view SQL statements in an active session.

By default, the onstat commands that show the SQL statement text from an active session are restricted to DBSA users. To remove this restriction, set the UNSECURE_ONSTAT configuration parameter to 1. The onstat commands that show SQL statements include onstat -g his, onstat -g ses, onstat -g stm, onstat -g ssc, and onstat -g sql.

The **UNSECURE_ONSTAT** configuration parameter takes effect when the database server is shut down and restarted.

Related information

[UNSECURE_ONSTAT configuration parameter on page](#)

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 HCL® 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, HCL® Informix® enforces two levels of access control. The first level is DAC. With DAC, HCL® 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.

Configuring label-based access control

The general procedure involves a few SQL-based tasks that define precise but flexible database security objects.

Before you begin

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.

About this task

The following list outlines the major tasks in setting up a basic implementation with HCL® 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.

Example

Tables to exclude from LBAC protection

What to do next

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.

Read Access Rules

When a user attempts to retrieve data from an LBAC-protected table with a SELECT operation, the comparison follows 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.

Write Access Rules

When a user attempts an INSERT, UPDATE, or DELETE operation, the comparison follows 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 HCL® 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 reference

[Requirements for trusted-context objects and trusted connections on page 100](#)

Related information

[Trusted-context objects and trusted connections on page 99](#)

[Creating a trusted-context object on page 101](#)

Granting the Database Security Administrator role

A DBSA uses the GRANT DBSECADM statement to give database security administrator authority to a user.

Before you begin

You must be a DBSA to grant DBSECADM.

About this task

Grant the DBSECADM role by issuing the GRANT DBSECADM statement, as described in the *HCL® Informix® Guide to SQL: Syntax*.

Example

The following statement gives DBSECADM authority to user `sam`:

```
GRANT DBSECADM TO sam;
```

What to do next

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.

Before you begin

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.

About this task

Use the REVOKE DBSECADM statement to revoke the DBSECADM role, as described in *HCL® Informix® Guide to SQL: Syntax*.

Example

The following statement revokes DBSECADM authority from user `sam`:

```
REVOKE DBSECADM FROM sam;
```

What to do next

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 141](#))
- **SET:** Each element represents one member of an unordered set (see [Security label component type: SET on page 142](#))
- **TREE:** Each element represents a node in a tree-like hierarchy (see [Security label component type: TREE on page 142](#))

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 142](#) 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.

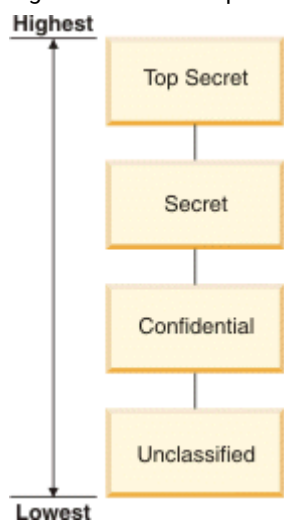


Example: If the fictional company JK Enterprises defines security label component `level` as a ranking of the company's four different privacy levels, as in the following statement:

```
CREATE SECURITY LABEL COMPONENT level
  ARRAY [ 'Top Secret', 'Secret', 'Confidential', 'Unclassified' ];
```

Then [Figure 9: Relationship of elements in an ARRAY example on page 141](#) illustrates the order of the elements:

Figure 9. 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.

Example

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 IDSLBACWRITASET 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.



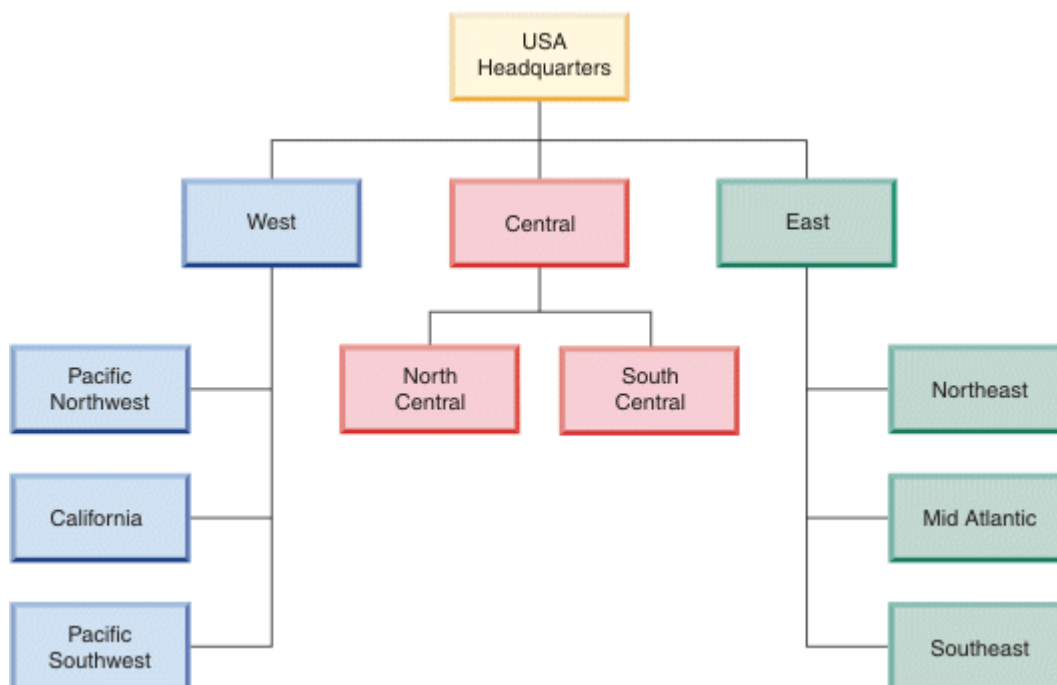
Example: JK Enterprises decides that its levels of authority to access protected data must follow its organizational chart. The company can use this scheme to outline its TREE security label component. The following example shows a statement creating the `region` security label component:



```
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 10: Relationship of elements in a TREE example on page 143 illustrates the relationships among the TREE component elements in this example.

Figure 10. 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:

- 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.

Before you begin

You must hold the DBSECADM role to create security label components.

About this task

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 *HCL® Informix® Guide to SQL: Syntax*.

Example

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.

Before you begin

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.

About this task

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 *HCL® Informix® Guide to SQL: Syntax*.

Example

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'};
```

What to do next

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 148](#) for how to attach the policy to a table.

Creating security policies

You create security policies after you have created security label components.

Before you begin

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.

About this task

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 *HCL® Informix® Guide to SQL: Syntax*.

Example

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 137](#).

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.

Before you begin

You must hold the DBSECADM role to create a security label.

About this task

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 *HCL® Informix® Guide to SQL: Syntax*.

Example

The following example shows a CREATE SECURITY LABEL statement:

```
CREATE SECURITY LABEL company.label2
  COMPONENT level 'Secret',
  COMPONENT function 'Administrative',
  COMPONENT region 'Southwest';
```

What to do next

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.

Before you begin

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.

About this task

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 *HCL® Informix® Guide to SQL: Syntax*

In the following example of this SQL statement, `label12` of the `company` security policy is granted to user `maria`.

```
GRANT SECURITY LABEL company.label12
TO maria;
```

What to do next

Revoking security labels

The REVOKE SECURITY LABEL statement revokes a security label from a user or from a list of users.

Before you begin

You must hold the DBSECADM role to issue the REVOKE SECURITY LABEL statement.

About this task

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 *HCL® Informix® Guide to SQL: Syntax*.

Example

In the following example of this SQL statement, `label12` of the `company` security policy is revoked from user `maria`.

```
REVOKE SECURITY LABEL company.label12
FROM maria;
```

What to do next

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 HCL® 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 *HCL® 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.

Example**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. IDSSECURITYLABEL 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 *HCL® Informix® Guide to SQL: Reference* and *HCL® 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 on page 150](#), [Applying column-level protection on page 150](#), *HCL® Informix® Guide to SQL: Reference*, and *HCL® 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.

Before you begin

About this task

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 *HCL® 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 *HCL® Informix® Guide to SQL: Syntax*.

Example

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;
```

What to do next

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.

Before you begin

About this task

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 *HCL® 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 *HCL® Informix® Guide to SQL: Syntax*.

Example

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 `label14`.

```
CREATE TABLE T3
(C1 CHAR (8),
 C2 int COLUMN SECURED WITH label14,
 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 `label15`.

```
ALTER TABLE T4
MODIFY (C1 CHAR (8) COLUMN SECURED WITH label15),
ADD SECURITY POLICY company;
```

What to do next

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.

Before you begin

You must hold the DBSECADM role to grant exemptions.

About this task

Grant an exemption by issuing the GRANT EXEMPTION statement, as described in the *HCL® Informix® Guide to SQL: Syntax*.

Example

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
```

What to do next

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.

Before you begin

You must hold the DBSECADM role to revoke exemptions.

About this task

Revoke an exemption by issuing the REVOKE EXEMPTION statement, as described in the *HCL® Informix® Guide to SQL: Syntax*.

Example

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
```

What to do next

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.

Run the `onstat -g cac lbacply` and `onstat -g cac lbacusr` commands to monitor the label-based access control (LBAC) caches.

Set the following configuration parameters to control the LBAC caches:

PLCY_HASHSIZE

Specifies the number of hash buckets in the security policy information cache.

PLCY_POOLSIZE

Specifies the maximum number of entries in each hash bucket of the security policy information cache.

USRC_HASHSIZE

Specifies the number of hash buckets in the LBAC credential memory cache.

USRC_POOLSIZE

Specifies the maximum number of entries in each hash bucket of the LBAC credential memory cache.

Tuning the LBAC caches

Poor performance of a database with tables protected by 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 LBAC configuration parameters in the `onconfig` file can improve performance for queries that are frequently run on protected tables. For example, if the value for the `PLCY_HASHSIZE` configuration parameter is set too low, there are not enough hash buckets that are allocated for security policy information caching and so some database performance that involves LBAC-protected tables declines.

Related information

[PLCY_HASHSIZE configuration parameter on page](#)

[PLCY_POOLSIZE configuration parameter on page](#)

[USRC_HASHSIZE configuration parameter on page](#)

[USRC_POOLSIZE configuration parameter on page](#)

[onstat -g cac command: Print information about caches on page](#)

Dropping security objects

Use the `DROP SECURITY` statement to remove a security label component, a security policy, or a security label from the database.

Before you begin

You must hold the `DBSECADM` role to remove a security object.

About this task

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
Choose from:
 - 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
Choose from:
 - Example: `DROP SECURITY LABEL company.label12` removes the label named `label12`.

3. DROP SECURITY LABEL COMPONENT *component* removes a security label component; this can be use in RESTRICT mode

Choose from:

- 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 *HCL® Informix® Guide to SQL: Syntax*.

Results

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:

What to do next

- **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.

Before you begin

You must hold the DBSECADM role to rename a security object.

About this task

The three valid clauses for the RENAME SECURITY statement are as follows:

1. POLICY *old_name* TO *new_name* renames a security policy

Choose from:

- 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

Choose from:

- Example: `RENAME SECURITY LABEL subsidiary.label18 TO label19;` renames `label18` to `label19`, which belongs to security policy `subsidiary`

3. LABEL COMPONENT *old_name* TO *new_name* specifies a security label component

Choose from:

- Example: `RENAME SECURITY LABEL COMPONENT department TO division;` renames the component `department` to `division`.

For more information about the RENAME SECURITY statement, see *HCL® Informix® Guide to SQL: Syntax*.

Results

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:

What to do next

- `syssecpolicies.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 `syssecpolicies.secpolicyid`, `sysseclabels.seclabelid`, or `sysseclabelcomponents.compoid` of the renamed security object.

HCL Informix® security considerations for label-based access control

The wide range of HCL® 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 on page 155](#))
- Exporting schema and data (see [The dbschema, dbexport, and dbimport Utilities on page 156](#))
- Importing data (see [The dbschema, dbexport, and dbimport Utilities on page 156](#))

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 156](#))
- [Data loading and unloading on page 156](#)

To prevent unauthorized access to protected tables, take extra precautions with the following database operations and objects:

- [The onlog utility on page 157](#)
- [The oncheck utility on page 157](#)
- [Enterprise replication on page 157](#)
- [Data definition language \(DDL\) operations on page 157](#)
- [INSERT INTO . . . SELECT FROM Statement on page 157](#)
- High Performance Loader .RET and .FLT files (see [Data loading and unloading on page 156](#))
- [Temporary tables created by the INTO TEMP clause on page 157](#)
- [User-defined routines on page 158](#) 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.

HCL® 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 HCL® 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 *HCL® 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

HCL® 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 HCL Informix® functionality with label-based access control

HCL® Informix® has non-security functionality that operates seamlessly with label-based access control.

HCL® 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 HCL® 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.

Example

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. HCL® 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

HCL® 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, HCL® 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.

Auditing data security

This section contains information about how to audit the security of your database.

Auditing with IBM® Security Guardium® (UNIX, Linux)

You can audit user actions for your Informix® database server with IBM® Security Guardium® version 10.0 or later. IBM® Security Guardium® prevents leaks from databases, ensures the integrity of information, and automates compliance controls across heterogeneous environments.

After you set up the IBM® Security Guardium® server, you start the ifxguard utility to monitor connections to your Informix® databases. You can set the logging mode and the number of worker threads to prevent heavy locking by editing the ifxguard configuration file. You can enable auditing and set the actions of the database server if the IBM® Security Guardium® server does not respond in the timeout period by setting the IFXGUARD configuration parameter in the `onconfig` file.

Every time a user session attempts an action that is auditable, an ifxguard agent contacts the IBM® Security Guardium® server. The IBM® Security Guardium® server can mask sensitive data or close a connection that fails the security audit.

To configure IBM® Security Guardium® and start auditing:

1. Install and configure IBM® Security Guardium®. See [IBM® Security Guardium® V 10.0](#).
2. Edit the `ifxguard` configuration file.
3. Set the IFXGUARD configuration parameter in your `onconfig` file for your Informix® database server.
4. Follow the instructions in the section Informix EXIT with UNIX S-TAP to configure and start the ifxguard utility. See [Installing an S-TAP on a UNIX server](#).

ifxguard configuration file

The `ifxguard` configuration file sets the properties of the ifxguard utility.

The default location of the `ifxguard` configuration file is `$INFORMIXDIR/etc/ifxguard.$INFORMIXSERVER`.

The `ifxguard` configuration file contains the following parameters:

NAME

The name of the ifxguard agent. Must be unique on the host computer.

WORKERS

The number of threads for the ifxguard agent. Default is 4. Each thread connects to the database server and processes client application communication. For better performance, set equal to or greater than the number of CPU virtual processors.

LIBPATH

The installation path of IBM® Security Guardium® library for Informix®. Default is `$INFORMIXDIR/lib/libguard_informix.so`.

DEBUG

Optional. The level of debugging:

- 0 = Default. Debugging is disabled.
- 1 = The ifxguard utility prints error messages.
- 2 = The ifxguard utility prints error messages, the routine name, and more debugging messages.

LOGFILE

Optional. The full path and file name for the ifxguard log file. If not set, messages are sent to `stdout`.

Example**Example**

For example, the following `ifxguard` configuration file starts 6 worker threads, and prints error messages to the log file:

```
NAME    ifxg_ol_informix1210_1
WORKERS 6
LIBPATH /usr/local/informix/lib/libguard_informix.so
DEBUG   2
LOGFILE /tmp/logfile.txt
```

ifxguard utility syntax

Use the ifxguard utility commands start, stop, and configure auditing.

Syntax

```
ifxguard [ { -p exit_file -l log_file_path [ -w workers ] } -c configuration_file | -x agent_name | -k agent_name }
```

Table 12. ifxguard utility syntax elements

Element	Purpose
<code>-c configuration_file</code>	Specifies the path and name of the <code>ifxguard</code> configuration file. Default is <code>\$INFORMIXDIR/etc/ifxguard.\$INFORMIXSERVER</code> .
<code>-k agent_name</code>	Stops the specified ifxguard agent.
<code>-l log_file_path</code>	Specifies the full path and file name for the ifxguard log file.

Table 12. ifxguard utility syntax elements (continued)

Element	Purpose
<code>-p exit_file</code>	Starts the <code>ifxguard</code> agent and generates a configuration file with the name <code>\$INFORMIXSERVER_guard</code> . Specifies the full path and file name of the Informix® EXIT file for IBM® Security Guardium®. Default is <code>\$HOME/lib/libguard_informix_exit_64.so</code> .
<code>-r agent_name</code>	Reloads the <code>ifxguard</code> configuration file for the specified <code>ifxguard</code> agent while the agent is running. Changes to only the debug level and the log file are recognized during reloading.
<code>-w workers</code>	Specifies the number of threads for the <code>ifxguard</code> agent.

Example**Example: Start the ifxguard agent with the default configuration file**

The following command starts the `ifxguard` agent with the configuration file in the default location:

```
ifxguard
```

Example**Example: Start the ifxguard agent and generate a configuration file**

The following command starts the `ifxguard` agent and generates a configuration file that specifies a log file and 6 worker threads:

```
ifxguard -p $HOME/lib/libguard_informix_exit_64.so -l /tmp/logfile.txt -w 6
```

Example**Example: Stop the ifxguard agent**

The following command stops the `ifxguard` agent that is named `ifxg_ol_informix1210_1`:

```
ifxguard -k ifxg_ol_informix1210_1
```

Example**Example: Change the configuration of the ifxguard agent**

The following command reloads the edited configuration file for the `ifxguard` agent that is named `ifxg_ol_informix1210_1`:

```
ifxguard -r ifxg_ol_informix1210_1
```

IFXGUARD configuration parameter

The IFXGUARD configuration parameter enables auditing with IBM® Security Guardium® and sets the actions of the database server if the IBM® Security Guardium® server does not respond in the timeout period.

onconfig.std value

Not present in the `onconfig.std` file

value if not present

```
IFXGUARD enable=1,timeout=-1
```

values

See the Usage section.

takes affect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the `onmode -wf` command.

When you reset the value in memory by running the `onmode -wm` command.

Usage

Use the IFXGUARD configuration parameter to control the behavior of the database server when a user session requires a response from IBM® Security Guardium®.

```
IFXGUARD { enable = 0 | enable = 1 timeout = { -1 | seconds : actions } }
```

Table 13. IFXGUARD configuration parameter options

Field	Purpose
enable	Specifies whether auditing is enabled: <ul style="list-style-type: none"> • 0 = Connections from the ifxguard agent are rejected and auditing is disabled. • 1 = Connections from the ifxguard agent are allowed and auditing is enabled.
timeout	Specifies the behavior when a user session is attempting an action that is audited:

Table 13. IFXGUARD configuration parameter options (continued)

Field	Purpose
	<ul style="list-style-type: none"> • -1 = The user session waits indefinitely for the ifxguard agent to get a response from the IBM® Security Guardium® server. • <i>seconds</i> = The user session waits for the number of seconds for the ifxguard agent to get a response from the IBM® Security Guardium® server and the database server takes the specified action.
<i>seconds</i>	The timeout period, in seconds, for the ifxguard agent to get a response from the IBM® Security Guardium® server.
<i>actions</i>	<p>Specifies the action for the database server to take after the ifxguard agent timeout period:</p> <ul style="list-style-type: none"> • ignore = The database server ignores the failure of the ifxguard agent and the user session continues. • alarm = The database server raises the event alarm 87003 for the timeout of the ifxguard utility and continues to wait for an ifxguard agent before it continues to process the user session connection. • kill = The database server shuts down the ifxguard agent and accepts the user session connection without auditing. IBM® Security Guardium® is disabled. • shutdown = The database server shuts down.

Example**Example**

The following entry enables IBM® Security Guardium® and specifies that if the ifxguard agent does not get a response from the IBM® Security Guardium® server within 5 seconds for the user session, the database server shuts down the ifxguard agent:

```
IFXGUARD enable=1,timeout=5,kill
```

Related information

[onmode -wf, -wm: Dynamically change certain configuration parameters on page](#)

[Event alarm IDs on page](#)

Overview of auditing with the Informix® secure auditing facility

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 *HCL® Informix® Backup and Restore Guide* explains these facilities.

Audit to Syslog facility (ASL)

In the classic auditing of Informix, audit log records are written to numbered files on the local file system. While the classic auditing continues to be supported, the audit subsystem for Informix on Unix-based systems has been modified to allow audit records to be sent to the syslog daemon, which can be configured to handle the messages in many different ways. Different systems have different syslog daemons with different capabilities and different ways of configuring them. To use ASL auditing, new options have been provided in `onaudit` and `onshowaudit` and new audit configuration parameters are introduced in the audit configuration file.

POSIX Syslog

The basic functionality used by Informix is defined by the POSIX standard. POSIX defines four functions – `openlog()`, `syslog()`, `closelog()`, and `setlogmask()` and one header `<syslog.h>`. POSIX does not define the behaviour or configuration of the syslog daemon itself; it only requires the functions (primarily `syslog()`) to report messages.

For more information, see [onaudit utility on page 214](#), [onshowaudit utility on page 221](#), and [the audit configuration file on page 250](#).

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. [Audit event codes and fields on page 225](#) 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.

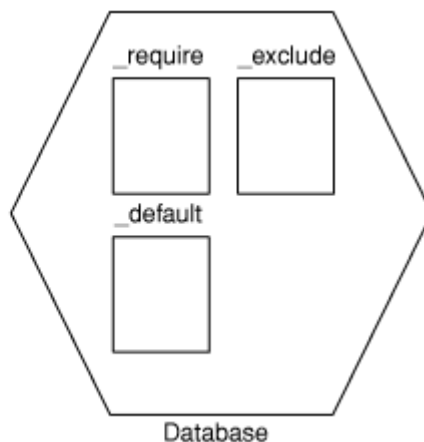
Before you use auditing, you must specify which audit events to audit. To specify audited events, add the events to the masks. You must also perform other tasks, which [Audit administration on page 185](#), describes.

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 [Audit analysis on page 203](#), describes.

[Figure 11: Audit masks after installation on page 165](#) represents a set of audit masks. The actual masks and their features are explained in [Audit masks and audit instructions on page 168](#).

Figure 11. Audit masks after installation

- After installation:
- Create audit masks
 - Turn on auditing



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 11: Audit masks after installation on page 165](#) 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 169](#).

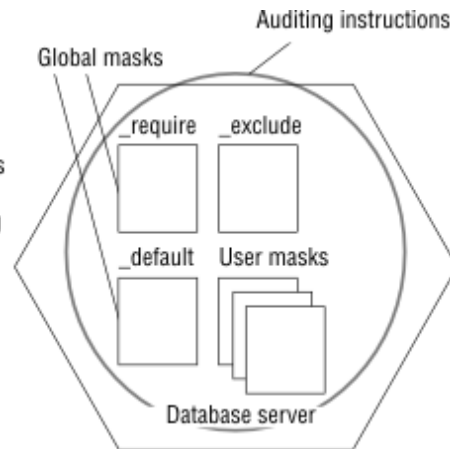
You can also create template masks for creating new user masks. For a description of template masks, see [Template masks on page 170](#).

Masks and their events are called auditing instructions, as [Figure 12: The auditing instructions on page 166](#) 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.

Figure 12. The auditing instructions

Defining masks:

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



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 [Audit administration on page 185](#).

Related reference

[The onaudit utility: Configure audit masks on page 210](#)

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 the 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 information

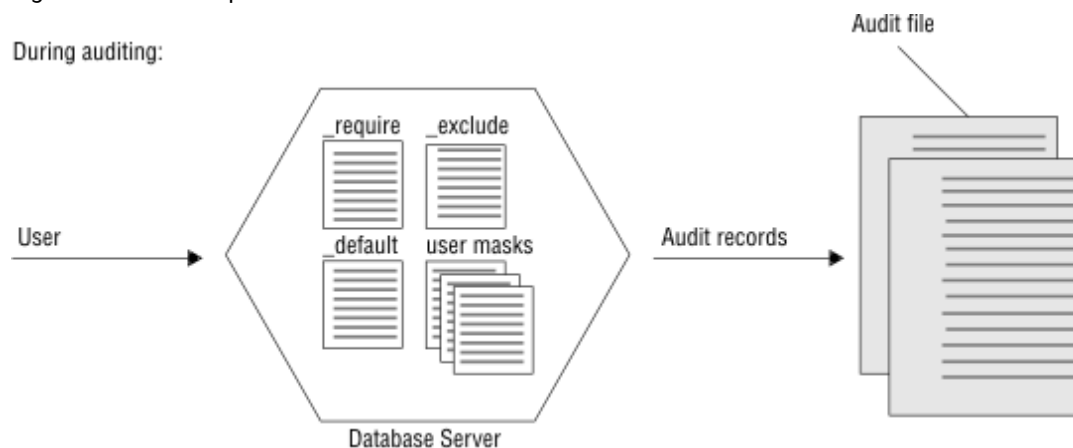
[Setting up selective row-level auditing on page 192](#)

Audit process

When you turn on auditing, the database server generates audit records for every event that the auditing instructions specify.

The database server stores the audit records in an audit file. The collection of audit records makes up the audit trail. (The audit trail might consist of more than one audit file.)

Figure 13. 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 173](#).

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.

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 179](#) 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 *HCL® 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: 2
- 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 187](#). For information about setting up role separation and creating a user group for each role, see your *HCL® Informix® Installation Guide*.

Audit masks and audit instructions

As described in [Audit masks on page 165](#), 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. [Audit event codes and fields on page 225](#)

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 170](#).


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 [Audit administration on page 185](#).

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 210](#).) Also, see the description of the `sysmaster` database in the *HCL® 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 [Audit event codes and fields on page 225](#).

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 *HCL® 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 [Audit administration on page 185](#) 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 [Audit administration on page 185](#), 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 210](#), 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. The classic configuration parameters are **`ADTERR`**, **`ADTMODE`**, **`ADTPATH`**, **`ADTROWS`**, and **`ADTSIZE`** of which `ADTMODE` is now deprecated.

For **`ASL`** (Audit to Syslog), there are additional configuration parameters which are added to the audit configuration file based on the options used with `onaudit` utility. These are **`ADT_CLASSIC_ENABLED`**, **`ADT_DBSSO`**, **`ADT_DBSA`**, **`ADT_SYSLOG_ENABLED`**, **`ADT_SYSLOG_IDENTIFIER`**, **`ADT_SYSLOG_OPTIONS`**, **`ADT_SYSLOG_FACILITY`**, and **`ADT_SYSLOG_PRIORITY`**.

For more information, see [the `ADTCFG` file on page 250](#).

The path name for the default `ADTCFG` file follows.

UNIX™

`$INFORMIXDIR/aaodir/adtcfg.servernum` if it exists, or

`$INFORMIXDIR/aaodir/adtcfg` if it does not.

Windows™

`%INFORMIXDIR%\aaodir\adtcfg.servernum` if it exists, or

`%INFORMIXDIR%\aaodir\adtcfg` if it does not.

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. However, the **ADTMODE** parameter is now deprecated; you should use **ADT_CLASSIC_ENABLED**, **ADT_DBSA**, and **ADT_DBSSO** instead.

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 *HCL® Informix® Administrator's Reference* describes). The configuration changes take effect in the server immediately. Henceforth, 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 214](#) and see [The `ADTCFG` file on page 250](#). For more information about auditing administration, see [Administrative roles and role separation on page 185](#).

Properties of audit files

As [Audit process on page 167](#) 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 191](#) 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 210](#).

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 [Audit analysis on page 203](#) and [The `onaudit` utility: Configure audit masks on page 210](#) 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 185](#).

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 their 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.

For more information about shared-memory connections, see the *HCL® 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.

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 *HCL® 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 [The onaudit utility on page 210](#).

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 the HCL® 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 210](#) 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 *HCL® Informix® Guide to SQL: Tutorial*. For information about this and other database server roles and users, see your *HCL® 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 188](#).

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_dbssso 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_dbssso**, 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 *HCL® Informix® Guide to SQL: Reference*. For more information about configuring role separation, see your *HCL® 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 [Overview of auditing with the Informix secure auditing facility on page 164](#).

Use the `onaudit` utility to add audit events to audit masks. [Audit event codes and fields on page 225](#) lists the audit events and their codes. [The `onaudit` utility: Configure audit masks on page 210](#) 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 210](#).

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 `-1 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 [The `ADTCFG` file on page 250](#).

For more information about the `onaudit` utility, see [The `onaudit` utility: Configure audit masks on page 210](#).

Related reference

[The `onaudit` utility: Configure auditing on page 214](#)

Setting the error mode

As [Overview of auditing with the Informix secure auditing facility on page 164](#) 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 210](#). For the `ADTERR` configuration parameter, see [The `ADTCFG` file on page 250](#).

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 214](#)

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 210](#). For information about the `ADTMODE` configuration parameter, see [The `ADTCFG` file on page 250](#).

 **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 .

Related reference

[The onaudit utility: Configure auditing on page 214](#)

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.

Before you begin

You must be a DBSSO to complete this task.

About 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.

Example

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 reference

[The onaudit utility: Configure auditing on page 214](#)

Related information

[Selective row-level auditing on page 166](#)

[ADD AUDIT Clause on page](#)

[DROP AUDIT Clause on page](#)

[Using the WITH AUDIT Clause on page](#)

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 214](#) and [The ADTCFG file on page 250](#).

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 202](#).

Related reference

[The onaudit utility: Configure auditing on page 214](#)

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

About this task

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 (_). [Overview of auditing with the Informix secure auditing facility on page 164](#) describes template masks and user masks.

Example

You specify events in the mask when you create it by using the audit events from the alphabetic listing in the table [Audit event codes and fields on page 225](#). 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

About this task

To create a template audit mask:

Use the `onaudit` utility.

Example

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
```

Related reference

[The `onaudit` utility: Configure audit masks on page 210](#)

Creating a user mask from a template mask

About this task

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.

Example

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

About this task

To create user masks without a template mask:

Use events as the basis for the user mask.

Example

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 210](#).

Adding one or more masks using an input file

About this task

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`.

Example

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 210](#).

```
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 173](#).) Specifically, audit masks are not automatically read from a file and initialized.

Related reference

[The onaudit utility: Configure audit masks on page 210](#)

Displaying audit masks

About this task

To display all the audit masks and the audit events that each mask contains:

Use the `-o` option of the `onaudit` utility.

Example

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 on page 198](#). For the complete syntax of the `onaudit -o` option and a description of the output, see [The onaudit utility: Configure audit masks on page 210](#).

 **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

About this task

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.

Example

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 210](#).

Deleting audit masks

About this task

To delete a single mask or all masks:

Use the `-d` option of the `onaudit` utility.

Example

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 210](#).

Audit configuration maintenance

About this task

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).

What to do next

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 210](#).

Displaying the audit configuration

About this task

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.

Example

```
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 **sysadinfo** table in the **sysmaster** database. For details, see the *HCL® Informix® Administrator's Reference*.

- On Windows™, the example shows output from the `onaudit -c` command.

Example

```
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
  ADTSIZE   = 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 214](#)

Starting a new audit file

About this task

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 [The onaudit utility on page 210](#).

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.

Example

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.

Example

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.

Example

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 214](#)

Changing audit levels

About this task

- Use the `onaudit` utility to change levels of auditing by the database server and to change the mandatory auditing of the DBSA.

Example

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:

Example

```
onaudit -l 5
```

Related reference

[The onaudit utility: Configure auditing on page 214](#)

Changing the audit error mode

About this task

As [Overview of auditing with the Informix secure auditing facility on page 164](#) and [Setting the error mode on page 191](#) 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.

Example

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 214](#)

Turning off auditing

About this task

To turn off auditing:

Use the `onaudit` utility.

Example

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 214](#)

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 [Overview of auditing with the Informix secure auditing facility on page 164](#).

Audit-record format

The database server generates the second part of the audit record, with fields that depend on the audit event.

[Table 14: Audit-record format on page 203](#) shows the format of the database server audit records.

Table 14. 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

Table 14. Audit-record format (continued)

ONLN	date and time	hostname or hostname. domain.ext	pid	database server name	user name	errno	event mnemonic	Additional Fields
...

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. [Audit event codes and fields on page 225](#) 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 210](#).

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 HCL® Informix® application development tools 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 179](#). 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 182](#).

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.

By default, tables that you create in databases that are not ANSI-compliant have privileges that allow access to all users. 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 *HCL® 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.

In a database that is not ANSI-compliant, you can also use the **NODEFDAC** environment variable to control access to tables and to UDRs. When set to **yes**, **NODEFDAC** prevents default table privileges (Select, Insert, Update, and Delete) from being granted to PUBLIC when a new table is created, and also prevents the Execute privilege from being granted by default to PUBLIC when a new UDR is created in Owner mode. For details, see the description of **NODEFDAC** in the *HCL® Informix® Guide to SQL: Reference*.

Preparing audit analysis records for SQL access

About this task

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

About this task

The first step to prepare for SQL-based audit analysis is to use `onshowaudit -I` to extract selected audit records in dbload format and put them in an output file.

Example

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 210](#).

Creating a database for audit data

About this task

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.

Example

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 205](#).

The following SQL statement creates an ANSI-compliant database:

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

Creating a table for audit data

About this task

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.

Example

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 209](#).

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 .

Creating a command file for dbload

About this task

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)

Example

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 206](#) and the `frag_logs` table created in [Creating a table for audit data on page 207](#).

```

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

```

What to do next

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

Loading audit data into a database

About this task

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.

Example

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 207](#):

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

What to do next

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 15: Audit-event columns in database table for SQL access on page 209](#) lists recommended column names that are used in [Creating a table for audit data on page 207](#) and describes the information that each column contains.

Table 15. 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)

Table 15. Audit-event columns in database table for SQL access (continued)

Column Name	Description
extra_1	Information specific to the object and event, as shown in Audit event codes and fields on page 225
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 Audit event codes and fields on page 225
extra_2	Information determined by the flag.

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

```
onaudit [{{ -m | -a } <Audit mask specification> | -f mask { basemask | - } <Audit mask specification> }][{ -o |
-a }][{ -usermask | -y }]
```

Audit mask specification

“ -umask ”

“ [-rbasemask] ”

“ -e [{ + | - }] { / event | r event | s event } ”

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 <i>basemask</i>	Specifies the name of an existing <i>basemask</i> from which you can derive events to apply to a new <i>targetmask</i> .	Subsequent changes to the <i>basemask</i> are not be reflected in the target audit masks. If no <i>basemask</i> 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 <i>targetmask</i> .	Events specified as arguments to -e override events listed in any base mask specified with the -r option.
-u <i>Fusermask</i>	Names a specific mask.	<i>_default</i> , <i>_require</i> , and <i>_exclude</i> 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.

Element	Purpose	Key Considerations
		The <i>usermask</i> is limited to 32 or fewer bytes.
-y	Automatically responds yes to the confirmation prompt.	None.
<i>event</i>	Specifies an event to audit, whether the event execution succeeds or fails.	The <i>event</i> must be listed in Audit event codes and fields on page 225 .
<i>Fevent</i>	Specifies that only failed event attempts are to be audited.	The <i>event</i> must be listed in Audit event codes and fields on page 225 .
<i>Sevent</i>	Specifies that only successful event attempts are to be audited.	The <i>event</i> must be listed in Audit event codes and fields on page 225 .

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.

Example

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

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

Example 3: Delete an audit mask

The following example shows how to delete the `_default` audit mask:

```
onaudit -d -u _default
```

Example**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 +GRXM, -e -CRTB
```

Example**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**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 information

[Audit masks on page 165](#)

[Adding one or more masks using an input file on page 196](#)

The onaudit utility: Configure auditing

Use the `onaudit` utility to start, stop, and configure auditing.

Syntax

```
>>-- onaudit --+-----+-----+-----+----->
                '-- -l --audit_mode--' '-- -e --error_mode--'
>--+-----+-----+-----+----->
```


Element	Purpose	Key Considerations
-n	Starts a new audit file.	You can use this option only when auditing is enabled.
-p <i>auditdir</i>	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.
-q	Suppresses the banner line which is written to standard error.	None
-s <i>maxsize</i>	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.
-A <i>flag</i>	Option for classic and ASL auditing. Enables or disables the mandatory auditing for the DBSA group. <ul style="list-style-type: none"> • 0, OFF, FALSE, DISABLE, NO = Disable mandatory auditing for the DBSA group • 1, ON, TRUE, ENABLE, YES = Enable mandatory auditing for the DBSA group 	This option sets the ADT_DBSA configuration parameter in the ADTCFG file.
-E <i>flag</i>	Option for ASL auditing. Enables or disables the Audit to Syslog (ASL) functionality. <ul style="list-style-type: none"> • 0, OFF, FALSE, DISABLE, NO = Turns ASL off • 1, ON, TRUE, ENABLE, YES = Turns ASL on 	This option sets the ADT_SYSLOG_ENABLED configuration parameter in the ADTCFG file.
-F <i>facility</i>	Option for ASL auditing.	This option sets the ADT_SYSLOG_FACILITY configuration parameter in the ADTCFG file.

Element	Purpose	Key Considerations
	<p>Helps with filtering messages in the syslog configuration.</p> <ul style="list-style-type: none"> • LOG_USER (the default) • LOG_LOCAL0..LOG_LOCAL7 • LOG_AUTH or LOG_AUTHPRIV <p>Other named facilities are for other subsystems and should not be used.</p> <p>The <i>facility</i> can be written with or without the LOG_prefix and in upper or lower-case or mixed case. The LOG_prefix and all upper-case is used when options are written to the ADTCFG file.</p>	
-l <i>identifier</i>	<p>Option for ASL auditing.</p> <p>Helps to choose the identifier name to be used in syslog messages. The maximum allowed length is 128 characters; the recommended maximum length is 32 characters.</p> <p>The default is the DBSERVERNAME from ONCONFIG file.</p>	<p>This option sets the ADT_SYSLOG_IDENTIFIER configuration parameter in the ADTCFG file.</p>
-L <i>flag</i>	<p>Option for classic auditing.</p> <p>Enables or disables classical (as opposed to syslog auditing) auditing.</p> <ul style="list-style-type: none"> • 0, OFF, FALSE, DISABLE, NO = Turn classic auditing off • 1, ON, TRUE, ENABLE, YES = Turn classic auditing on 	<p>This option sets the ADT_CLASSIC_ENABLED configuration parameter in the ADTCFG file.</p>
-O <i>options</i>	<p>Option for ASL auditing.</p> <p>Specifies options to openlog().</p>	<p>This option sets the ADT_SYSLOG_OPTIONS configuration parameter in the ADTCFG file.</p>

Element	Purpose	Key Considerations
	<ul style="list-style-type: none"> • LOG_NDELAY, LOG_NOWAIT = the default option • LOG_NDELAY, LOG_ODELAY = mutually exclusive options • LOG_PERROR, LOG_CONS, LOG_PID <p>The <i>options</i> can be written with or without the LOG_prefix and in upper or lower-case or mixed case. The LOG_prefix and all upper-case is used when options are written to the ADTCFG file.</p>	
-P <i>priority</i>	<p>Option for ASL auditing.</p> <p>Specify a priority while filtering messages in the syslog daemon.</p> <ul style="list-style-type: none"> • LOG_INFO • LOG_NOTICE • LOG_WARNING • LOG_DEBUG • LOG_ALERT • LOG_EMERG -should not be used. <p>The <i>priority</i> can be written with or without the LOG_prefix and in upper or lower-case or mixed case. The LOG_prefix and all upper-case is used when options are written to the ADTCFG file.</p>	<p>This option sets the ADT_SYSLOG_PRIORITY configuration parameter in the ADTCFG file.</p>
-R <i>row_mode</i>	<p>Controls selective row-level auditing:</p> <ul style="list-style-type: none"> • 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. 	<p>This option sets the ADTROWS configuration parameter in the ADTCFG file.</p>
-S <i>flag</i>	<p>Option for ASL auditing.</p>	<p>This option sets the ADT_DBSSO configuration parameter in the ADTCFG file.</p>

Element	Purpose	Key Considerations
	Enables or disables the mandatory auditing for the DBSSO group.	<ul style="list-style-type: none"> • 0, OFF, FALSE, DISABLE, NO = Disable mandatory auditing for the DBSSO group • 1, ON, TRUE, ENABLE, YES = Enable mandatory auditing for the DBSSO group

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 option letters of this utility must be entered as shown because they are case-sensitive.

The `onaudit` command takes effect immediately for all new 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.

`onaudit -h` output:

```

onaudit <action> [-q] [-f file] [-u name] [-r bmsk] [-e eset] [-y]
onaudit [-h] [-q] [-c] [-n] [-l lev] [-e err] [-p path] [-s size] \
        [-R fga] [-E {on|off}] [-F facility] [-I identifier] \
        [-O options] [-P priority] [-L level] [-A {on|off}] \
        [-S {on|off}]

-h          -- print help message and exit
-q          -- quiet mode

DBSSO options:
action: one of
  -a        -- add a mask
  -d        -- delete a mask
  -m        -- modify a mask
  -o        -- output a mask
  -e eset   -- event set added to (+) or removed from (-) mask
  -f file   -- include instruction file
  -r bmsk   -- name of basemask
  -u mask   -- name of target/mask
  -y        -- respond yes to all prompts

DBSA options:
  -c        -- print audit configuration
  -e err    -- set ADTERR
  -l lev    -- set ADTMODE (obsolescent: use -A, -L, -S)
  -n        -- start new log file
  -p path   -- set ADTPATH

```

```

-s size      -- set ADTSIZE
-A flag      -- enable/disable mandatory auditing of DBSA
-L flag      -- enable/disable classic audit
-R flag      -- set ADTROWS for Fine-Grained Auditing (0,1,2)
-S flag      -- enable/disable mandatory auditing of DBSSO
               (NB: The -A, -L, -S options supersede obsolescent -l option.)

ASL (Audit-to-Syslog) options:
-E flag      -- Enable/disable Audit-to-Syslog (ASL) (0,1, true, false, on, off)
-F facility  -- Set ASL facility (default: LOG_USER):
               (suggested: LOG_USER, LOG_LOCAL0..LOG_LOCAL7, LOG_AUTH,
                LOG_AUTHPRIV;
                not recommended: LOG_CRON, LOG_DAEMON, LOG_FTP, LOG_KERN,
                LOG_LPR, LOG_MAIL, LOG_NEWS, LOG_SYSLOG, LOG_UUCP)
-I identity  -- Set ASL identity (default: DBSERVERNAME)
-O options   -- Set ASL options (default: LOG_NDELAY, LOG_NOWAIT):
               (LOG_CONS, LOG_NDELAY, LOG_ODELAY, LOG_NOWAIT, LOG_PERROR,
                LOG_PID)
-P priority  -- Set ASL priority (aka level; default: LOG_INFO):
               (LOG_EMERG, LOG_ALERT, LOG_CRIT, LOG_ERR, LOG_WARNING,
                LOG_NOTICE, LOG_INFO, LOG_DEBUG)

```

The distributed `adtcfg` and `adtcfg.std` template files contain `ADT_ENABLED`, `ADT_DBSA`, `ADT_DBSSO` settings, and only mention `ADTMODE` in comments.

Example

Example 1: Start auditing

The following command starts classic auditing all sessions specified by audit masks (without mandatory auditing for DBSA or DBSSO users):

```
onaudit -L 1
```

Example

Example 2: Stop auditing

The following command stops classic auditing for sessions started after the command is:

```
onaudit -L 0
```

Example

Example 3: Change the audit configuration

The following command changes the error mode to 3 (shut down the server), the auditing mode to 3 (shut down the server if an error occurs while writing audit log records), enables classic auditing, sets the mandatory DBSSO auditing mode on, and starts a new audit file:

```
onaudit -e 3 -n -L 1 -S 1
```

Example

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
```

Example

Example 5: Enable Audit to Syslog

The following command enables ASL auditing and enables both, the mandatory DBSSO auditing mode and the mandatory DBSA auditing mode, without changing whether classic auditing is enabled. Note that the mandatory auditing affects both classic and ASL auditing.

```
onaudit -E on -S on -A on
```

Related reference

[The ADTCFG file on page 250](#)

Related information

[Setting up selective row-level auditing on page 192](#)

[Specifying a directory for the audit trail \(UNIX\) on page 190](#)

[Setting the error mode on page 191](#)

[Setting the audit level on page 191](#)

[Activating auditing on page 194](#)

[Displaying the audit configuration on page 199](#)

[Starting a new audit file on page 200](#)

[Changing audit levels on page 201](#)

[Changing the audit error mode on page 202](#)

[Turning off auditing on page 202](#)

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™:


```
>>-- onshowaudit --+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+----->
                    '-- -d --' '-- -q --' '-- -n --servernumber--' '-- -f --path--'
>--+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----<
```

```
'- -u --username--' '- -s --servername--' '- -l --loadfile--'
>>- onshowaudit -- -h --<
```

Windows™:

```
>>-onshowaudit--++-----+-----+-----+-----+-----+----->
      '- -n--servername-' '- -f--path-' +- -ts-+
                          '- -tf-'

>--++-----+-----+-----+-----+-----+-----><
+- -u--username-- -s--servername+ '- -l--loadfile-'
'- -d-----'
```

Element	Purpose	Key Considerations
-d	The output is not double-spaced when this option is used on POSIX systems. By default, there is a blank line after each audit record; the -d option omits that line.	On POSIX, it is used to avoid double-spacing in the output.
-f <i>path</i>	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 <i>path</i> is only a file name, see the notes that immediately follow this table.
-h	Prints a help message, the command line summary and an explanation of the options.	None
-l	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 (and selected with the -O option). The -l option may be used for compatibility, but may safely be omitted. No error will appear if someone uses -l, but it won't affect the operation of the command.	This option is case-sensitive. The UNIX™ operating system uses the Informix® database server audit trail.
-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.

Element	Purpose	Key Considerations
		<p>On the Windows™ operating system, you must enter a load file name argument for the -l option.</p> <p>On the UNIX operating system this file name argument is optional.</p> <p>On the UNIX™ operating system, if you do not specify a file name, the output is routed to standard output.</p>
-n <i>servername</i>	Extracts audit records from the ADTPATH location specified in the <i>adtcfg.servernumber</i> file.	If the <i>adtcfg.servernumber</i> file does not exist, the contents of the ADTCFG file are used for audit configuration.
-q	Suppresses the banner line and the 'program over' message when the program completes.	None
-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 <i>servername</i>	Specifies which database server must have audit information extracted.	None.
-u <i>username</i>	Specifies the login name of a user for extraction of audit information.	None.

**Note:**

-d option, when used on windows, indicates that the onshowaudit utility must use default values for the user (**current user**) and database server (**INFORMIXSERVER**) fields.

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 the 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.



Note:

The onshowaudit utility can be used to filter the audit data reported to the syslog daemon using ASL, but there are two issues: one, the syslog daemon adds information at the start of a line compared to classic audit logs; and two, the locations and names of the files recorded by the syslog daemon are not known to Informix and may not even be on the current machine.

Example 1 – Classic log (1 line):

```
ONLN|2021-01-12 18:59:21.512|njdc-ldev04|31055|njdc_ldev04_11|someone|0:ACTB:stores:someone:stock:110
```

Example 2 – syslog log (1 line):

```
Jan 12 11:59:21 njdc-ldev04 njdc_ldev04_11: ONLN|2021-01-12
18:59:21.512|njdc-ldev04|31055|njdc_ldev04_11|someone|0:ACTB:stores:someone:stock:110
```




The data in example 2 has 5 fields before the normal start of the Informix record (which begins at “ONLN|?”). (The machine/host name is njdc-ldev04; the Informix server name is njdc_ldev04_11. The default identifier is used here.)

However, because there are no pipe symbols in the prefixed information, you can use:

```
onshowaudit -f /var/log/informix.audit
```

(possibly with filters such as ‘-u someone’) to select data from the file. The output will include the data prefixed by the syslog daemon. (You might note that the syslog daemon above is running in time zone UTC-07:00, but the server is running in time zone UTC0 – hence the difference of 7 hours between the timestamp from the syslog daemon and the auditing code. Actually, one can’t tell which time zones are in use, but can tell that they are 7 hours apart.)

Example

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

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

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

[The ADTCFG file on page 250](#)

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 16: Audit events listed by event code on page 226 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 on page 245](#) 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 16. Audit events listed by event code

Event Code	Event	Variable Contents
ACTB	Access Table	dbname: <i>database_name</i> tabid: <i>owner_name, table_id</i>
ADCK	Add Chunk	dbname: <i>dbspace, name</i> extra_1: <i>offset</i> flags: <i>mirror_status</i> ¹ extra_2: <i>path and size</i>
ADLG	Add Transaction Log	dbname: <i>dbspace, name</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		extra_1: <i>log_size</i>
ALFR	Alter Fragement	dbname: <i>database_name</i> tabid: <i>table_id</i> objname: <i>index_name</i> extra_1: <i>operation_type</i> ¹⁸ login: <i>owner</i> flags: <i>frag_flags</i> ¹⁵ extra_2: <i>dbspaces</i> alter_type: 0 = normal, 1 = forced alter
ALIX	Alter Index	dbname: <i>database_name</i> tabid: <i>table_id</i> login: <i>owner</i> ¹⁴ flags: <i>cluster_flag</i> ^{9,14} extra_2: <i>index_name</i> ¹⁴
ALLC	Alter Security Label Component	dbname: <i>database_name</i> objname: <i>component_name</i> extra_2: <i>component_type</i>
ALME	Alter Access Method	dbname: <i>database_name</i> tabid: <i>access, method_ID</i> objname: <i>access_method, name</i> login: <i>access_method, owner</i>
ALOC	Alter Operator Class	dbname: <i>database_name</i> extra_1: <i>cluster_size</i> login: <i>owner</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		<i>extra_2: cluster_name</i>
ALOP	Alter Optical Cluster	<i>dbname: database_name</i> <i>extra_1: cluster_size</i> <i>login: owner</i> <i>extra_2: cluster_name</i>
ALSQ	Alter Sequence	<i>dbname: database_name</i> <i>tabid: table_id</i>
ALTB	Alter Table	<i>dbname: database_name</i> <i>tabid: old_table_id</i> <i>extra_1: new_table_id¹⁴</i> <i>partno: frag_id</i> <i>extra_2: new_part_number_list¹⁴</i>
ALTX	Alter trusted context	<i>dbname: database_name</i> <i>objname: context_name</i> <i>login: system_authid</i>
ALUR	Alter User	<i>objname: user_name</i>
BGTX	Begin Transaction	none
CLDB	Close Database	<i>dbname: database_name</i>
CMTX	Commit Transaction	none
CRAG	Create Aggregate	<i>dbname: database_name</i> <i>objname: aggregate_name</i> <i>login: owner</i>
CRAM	Create Audit Mask	<i>login: user_id</i>
CRBS	Create Storage Space	<i>dbname: storage_name, space_name</i> <i>login: owner</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		flags: <i>mirror_status</i> ¹ extra_2: <i>media</i>
CRBT	Create Opaque Type	dbname: <i>database_name</i> objname: <i>opaque_type_name</i> login: <i>opaque_type</i> , <i>owner</i>
CRCT	Create Cast	dbname: <i>database_name</i> tabid: <i>from_type_ID</i> objname: <i>function_name</i> or "-" extra_1: <i>from_type_xid</i> partno: <i>to_type_ID</i> row_num: <i>to_type_xid</i> login: <i>function_owner</i> or "-"
CRDB	Create Database	dbname: <i>dbspace</i> extra_2: <i>database_name</i>
CRDS	Create Dbspace	dbname: <i>dbspace</i> , <i>name</i> flags: <i>mirror_status</i> ¹
CRDT	Create Distinct Type	dbname: <i>database_name</i> objname: <i>distinct_type_name</i> login: <i>distinct_type</i> , <i>owner</i>
CRIX	Create Index	dbname: <i>database_name</i> tabid: <i>table_id</i> objname: <i>index_name</i> login: <i>owner</i> flags: <i>frag_flags</i> ¹⁵

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		<i>extra_2: dbspace_list</i>
CRLB	Create Security Label	dbname: <i>database_name</i> objname: <i>policy.label_name</i>
CRLC	Create Security Label Component	dbname: <i>database_name</i> objname: <i>component_name</i>
CRME	Create Access Method	dbname: <i>database_name</i> tabid: <i>access_method_ID</i> objname: <i>access_method_name</i> login: <i>access_method_owner</i>
CROC	Create Operator Class	dbname: <i>database_name</i> tabid: <i>operator_class_ID</i> objname: <i>operator_class_name</i> login: <i>owner</i>
CROP	Create Optical Cluster	dbname: <i>database_name</i> tabid: <i>table_id</i> <i>extra_1: cluster_size</i> login: <i>owner</i> <i>extra_2: cluster_name</i>
CRPL	Create Security Policy	dbname: <i>database_name</i> objname: <i>policy_name</i>
CRPT	Decryption Failure or Attempt	dbname: <i>database_name</i> objname: <i>statement</i>
CRRL	Create Role	dbname: <i>database_name</i> objname: <i>rolename</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
CRRT	Create Named Row Type	dbname: <i>database_name</i> tabid: <i>row_type_xid</i> objname: <i>named_row_type_name</i> login: <i>named_row_type_owner</i>
CRSN	Create Synonym	dbname: <i>database_name</i> tabid: <i>synonym_table_id</i> extra_1: <i>base_table_id</i> login: <i>owner</i> flags: <i>synonym_type</i> ⁷ extra_2: <i>synonym_name</i>
CRSP	Create SPL Routine	dbname: <i>database_name</i> tabid: <i>proc_id</i> login: <i>owner</i> extra_2: <i>procedure_name</i>
CRSQ	Create Sequence	dbname: <i>database_name</i> tabid: <i>table_id</i> objname: <i>owner</i>
CRTB	Create Table	dbname: <i>database_name</i> tabid: <i>table_id</i> objname: <i>owner</i> login: <i>table_name</i> flags: <i>frag_flags</i> ¹⁵ extra_2: <i>dbspace_list</i>
CRTTR	Create Trigger	dbname: <i>database_name</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		tabid: <i>table_id</i>
		row_num: <i>trigger_id</i> ¹⁴
		login: <i>owner</i> ¹⁴
		extra_2: <i>trigger_name</i> ¹⁴
CRTX	Create trusted context	dbname: <i>database_name</i>
		objname: <i>context_name</i>
		login: <i>system_authorization_id</i>
CRUR	Create User	objname: <i>user_name</i>
CRVW	Create View	dbname: <i>database_name</i>
		tabid: <i>view_table_id</i>
		login: <i>owner</i>
		extra_2: <i>view_name</i>
CRXD	Create XADatasource	dbname: <i>database_name</i>
		objname: <i>owner</i>
		objname: <i>XA_data_source_name</i>
CRXT	Create XADatasource Type	dbname: <i>database_name</i>
		objname: <i>owner</i>
		objname: <i>XA_data_source_type_name</i>
DLRW	Delete Row	dbname: <i>database_name</i>
		tabid: <i>table_id</i>
		extra_1: <i>part_number</i>
		partno: <i>frag_id</i>
		row_num: <i>row_number</i> ¹⁴
DNCK	Bring Chunk Offline	extra_1: <i>chunk_number</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		flags: <i>mirror_status</i> ¹
DNDM	Disable Disk Mirroring	extra_1: <i>dbspace_number</i>
DRAG	Drop Aggregate	dbname: <i>database_name</i> objname: <i>aggregate_name</i> login: <i>owner</i>
DRAM	Delete Audit Mask	login: <i>user_id</i>
DRBS	Drop Storage Space	dbname: <i>storage_space_name</i>
DRCK	Drop Chunk	dbname: <i>dbspace_name</i> flags: <i>mirror_status</i> ¹ extra_2: <i>path</i>
DRCT	Drop Cast	dbname: <i>database_name</i> tabid: <i>from_type_ID</i> extra_1: <i>from_type_xid</i> partno: <i>to_type_ID</i> row_num: <i>to_type_xid</i>
DRDB	Drop Database	dbname: <i>database_name</i>
DRDS	Drop Dbspace	dbname: <i>dbspace_name</i>
DRIX	Drop Index	dbname: <i>database_name</i> tabid: <i>table_id</i> login: <i>owner</i> extra_2: <i>index_name</i>
DRLB	Drop Security Label	dbname: <i>database_name</i> objname: <i>policy.label_name</i>
DRLC	Drop Security Label Component	dbname: <i>database_name</i> objname: <i>component_name</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
DRLG	Drop Transaction Log	extra_1: <i>log_number</i>
DRME	Drop Access Method	dbname: <i>database_name</i> tabid: <i>access_method_ID</i> objname: <i>access_method_name</i> login: <i>access_method_owner</i>
DROC	Drop Operator Class	dbname: <i>database_name</i> objname: <i>operator_class_name</i> login: <i>owner</i>
DROP	Drop Optical Cluster	dbname: <i>database_name</i> login: <i>owner</i> extra_2: <i>cluster_name</i>
DRPL	Drop Security Policy	dbname: <i>database_name</i> objname: <i>policy_name</i>
DRRL	Drop Role	dbname: <i>database_name</i> objname: <i>role_name</i>
DRRT	Drop Named Row Type	dbname: <i>database_name</i> tabid: <i>dropped_type_xid</i>
DRSN	Drop Synonym	dbname: <i>database_name</i> tabid: <i>synonym_table_id</i> login: <i>owner</i> extra_2: <i>synonym_name</i>
DRSP	Drop SPL Routine	dbname: <i>database_name</i> login: <i>owner</i> extra_2: <i>spname</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
DRSQ	Drop Sequence	dbname: <i>database_name</i> tabid: <i>table_id</i>
DRTB	Drop Table	dbname: <i>database_name</i> tabid: <i>table_id</i> objname: <i>table_name</i> login: <i>owner</i> flags: <i>drop_flags</i> ²¹ extra_2: <i>part_number_list</i>
DRTR	Drop Trigger	dbname: <i>database_name</i> row_num: <i>trigger_id</i> login: <i>owner</i> extra_2: <i>trigger_name</i>
DRUR	Drop User	objname: <i>user_name</i>
DRTX	Drop trusted context	objname: <i>context_name</i>
DRTY	Drop Type	dbname: <i>database_name</i> objname: <i>type_name</i> login: <i>type_owner</i>
DRVW	Drop View	dbname: <i>database_name</i> tabid: <i>view_table_id</i> flags: <i>drop_flags</i> ²¹
DRXD	Drop XADatasource	dbname: <i>database_name</i> objname: <i>owner</i> objname: <i>XA_data_source_name</i>
DRXT	Drop XADatasource Type	dbname: <i>database_name</i> objname: <i>owner</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		objname: <i>XA_data_source_type_name</i>
EXSP	Execute SPL Routine	dbname: <i>database_name</i> tabid: <i>proc_id</i>
GRDB	Grant Database Access	dbname: <i>database_name</i> extra_1: <i>privilege</i> ⁵ extra_2: <i>grantees</i> ⁴
GRDR	Grant Default Role	dbname: <i>database_name</i> objname: <i>role_name</i> login: <i>grantor</i> extra_2: <i>grantees</i> ⁴
GRFR	Grant Fragment Access	dbname: <i>database_name</i> tabid: <i>table_id</i> objname: <i>fragment</i> extra_1: <i>privilege</i> ^{5,14} login: <i>grantor</i> extra_2: <i>grantees</i> ^{4,14}
GRLB	Grant Security Label	dbname: <i>database_name</i> objname: <i>policy.label_name</i> login: <i>grantee</i> ⁴ extra_2: <i>access_type</i>
GRRL	Grant Role	dbname: <i>database_name</i> objname: <i>role_name</i> login: <i>grantor</i> extra_2: <i>grantees</i> ⁴

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
GRSA	Grant DBSECADM	login: <i>grantee</i>
GRSS	Grant SETSESSIONAUTH	dbname: <i>database_name</i> login: <i>grantee</i> extra_2: <i>surrogate_user_list</i>
GRTB	Grant Table Access	dbname: <i>database_name</i> tabid: <i>table_id</i> extra_1: <i>privilege</i> ^{5, 14} login: <i>grantor</i> extra_2: <i>grantee</i> ^{4, 14} , <i>update_columns</i> , <i>select_columns</i> ^{4, 14}
GRXM	Grant Exemption	dbname: <i>database_name</i> objname: <i>policy_name</i> login: <i>grantee</i> extra_2: <i>rule</i>
INRW	Insert Row	dbname: <i>database_name</i> tabid: <i>table_id</i> partno: <i>frag_id</i> row_num: <i>row_id</i>
LGDB	Change Database Log Mode	dbname: <i>database_name</i> flags: <i>log_status</i> ⁶
LKTB	Lock Table	dbname: <i>database_name</i> tabid: <i>table_id</i> flags: <i>lock_mode</i> ⁸
LSAM	List Audit Masks	none
LSDB	List Databases	none

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
MDLG	Modify Transaction Logging	flags: <i>buffered_log_flags</i> ²
ONAU	onaudit	extra_2: <i>command_line</i>
ONBR	onbar	extra_2: <i>command_line</i>
ONCH	oncheck	extra_2: <i>command_line</i>
ONIN	oninit	extra_2: <i>command_line</i>
ONLG	onlog	extra_2: <i>command_line</i>
ONLO	onload	extra_2: <i>command_line</i>
ONMN	onmonitor	extra_2: <i>command_line</i>
ONMO	onmode	extra_2: <i>command_line</i>
ONPA	onparams	extra_2: <i>command_line</i>
ONPL	onpload	extra_2: <i>command_line</i>
ONSP	onspaces	extra_2: <i>command_line</i>
ONST	onstat	extra_2: <i>command_line</i>
ONTP	ontape	extra_2: <i>command_line</i>
ONUL	onunload	extra_2: <i>command_line</i>
OPDB	Open Database	dbname: <i>database_name</i> flags: <i>exclusive_flag</i> extra_2: <i>database_password</i>
OPST	Optimize Storage	fragment <parameters>: <i>part_numbers</i> table <parameters>: <i>table_name:database_name:owner_name</i> compression purge_dictionary: <i>date</i>
PWUR	Set User Password	objname: <i>user_name</i>
RBSV	Rollback to Savepoint	dbname: <i>database_name</i> extra_1: <i>transaction_id</i> objname: <i>savepoint_name</i>
RDRW	Read Row	dbname: <i>database_name</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		tabid: <i>table_id</i>
		extra_1: <i>part_number</i>
		partno: <i>frag_id</i>
		row_num: <i>row_id</i> ¹⁴
RLOP	Release Optical Cluster	dbname: <i>family_name</i>
		row_num: <i>volume_number</i>
RLSV	Release Savepoint	dbname: <i>database_name</i>
		extra_1: <i>transaction_id</i>
		objname: <i>savepoint_name</i>
RLTX	Rollback Transaction	none
RMCK	Clear Mirrored Chunks	extra_1: <i>dbspace_number</i>
RNUR	Rename User	objname: <i>old_user_name</i>
		extra_2: <i>new_user_name</i>
RNDB	Rename Database	dbname: <i>database_name</i>
		objname: <i>new_dbname</i>
		login: <i>user_id</i>
RNDS	Rename dbspace	dbname: <i>dbspace_name</i>
		objname: <i>new_dbspace_name</i>
RNIX	Rename Index	dbname: <i>index_name</i>
		objname: <i>new_index_name</i>
RNLB	Rename Security Label	dbname: <i>database_name</i>
		objname: <i>old_policy.label_name</i>
		extra_2: <i>new_policy.label_name</i>
RNLC	Rename Security Label Component	dbname: <i>database_name</i>
		objname: <i>old_component_name</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		<i>extra_2: new_component_name</i>
RNPL	Rename Security Policy	<i>dbname: database_name</i> <i>objname: old_policy_name</i> <i>extra_2: new_policy_name</i>
RNSQ	Rename Sequence	<i>dbname: database_name</i> <i>tabid: table_id</i>
RNTC	Rename Table/Column	<i>dbname: database_name</i> <i>tabid: table_id</i> <i>objname: new_table/column_name</i> <i>extra_1: colno(*)</i> <i>login: owner</i> <i>extra_2: table_name(**)</i>
RNTX	Rename trusted context	<i>objname: context_name</i> <i>extra_2: new_context name</i>
RSOP	Reserve Optical Cluster	<i>dbname: family_name</i> <i>row_num: volume_number</i>
RVDB	Revoke Database Access	<i>dbname: database_name</i> <i>extra_1: privilege⁵</i> <i>extra_2: revokees⁴</i>
RVDR	Revoke Default Role	<i>dbname: database_name</i> <i>objname: role_name</i> <i>login: revoker</i> <i>extra_2: revokees⁴</i>
RVFR	Revoke Fragment Access	<i>dbname: database_name</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		tabid: <i>table_id</i>
		objname: <i>fragment</i>
		extra_1: <i>privilege</i> ^{5, 14}
		login: <i>revoker</i>
		extra_2: <i>revokees</i> ^{4, 14}
RVLB	Revoke Security Label	dbname: <i>database_name</i>
		objname: <i>policy.label_name</i>
		login: <i>grantee</i>
		extra_2: <i>access_type</i>
RVRL	Revoke Role	dbname: <i>database_name</i>
		objname: <i>role_name</i>
		login: <i>revoker</i>
		extra_2: <i>revokees</i> ⁴
RVSA	Revoke DBSECADM	login: <i>grantee</i>
RVSS	Revoke SETSESSIONAUTH	dbname: <i>database_name</i>
		login: <i>grantee</i>
		extra_2: <i>surrogate_user_list</i>
RVTB	Revoke Table Access	dbname: <i>database_name</i>
		tabid: <i>table_id</i>
		extra_1: <i>privilege</i> ^{5, 14}
		login: <i>revoker</i>
		flags: <i>drop_flags</i> ²¹
		extra_2: <i>revokees</i> ^{4, 14}
RVXM	Revoke Exemption	dbname: <i>database_name</i>
		objname: <i>policy_name</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		login: <i>grantee</i>
		extra_2: <i>rule</i>
SCSP	System Command, SPL Routine	extra_2: <i>command_string</i>
STCO	Set Collation	dbname: <i>database_name</i> objname: <i>locale_name</i>
STCN	Set Constraint	dbname: <i>database_name</i> flags: <i>constraint_mode</i> ¹¹ extra_2: <i>constraint_names</i>
STDF	Set Debug File	dbname: <i>database_name</i> extra_2: <i>file_path</i>
STDP	Set Database Password	dbname: <i>database_name</i> login: <i>user_id</i>
STDS	Set Dataskip	flags: <i>skip flags</i> ¹⁶ extra_2: <i>dbspace_list</i>
STEP	Set Encryption Password	dbname: <i>database_name</i>
STEV	Set Environment	objname: <i>environment_variable_and_value</i>
STEX	Set Explain	flags: <i>explain_flags</i> ¹²
STIL	Set Isolation Level	extra_1: <i>isolation_level</i> ³
STLM	Set Lock Mode	flags: <i>wait_flags</i> ¹³
STNC	Set No Collation	dbname: <i>database_name</i> objname: <i>locale_name</i>
STOM	Set Object Mode	dbname: <i>database_name</i> tabid: <i>table_id</i> extra_1: <i>command_mode_flag</i> ²²

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		flags: <i>object_type_flag</i> ²³ extra_2: <i>object_names</i>
STOP	Stop Violations	dbname: <i>database_name</i> tabid: <i>table_id</i>
STPR	Set Pdqpriority	flags: <i>priority_level</i> ¹⁷
STRL	Set Role	dbname: <i>database_name</i> objname: <i>role_name</i>
STRS	Set Resident	dbname: <i>database_name</i> objname: <i>fragment_list</i> extra_1: <i>fragment_information</i>
STRT	Start Violations	dbname: <i>database_name</i> tabid: <i>table_id</i> extra_1: <i>Vio_tid</i> flags: <i>Dia_tid</i>
STSA	Set Session Authorization	dbname: <i>database_name</i> login: <i>new_user_name</i>
STSC	Set Statement Cache	objname: <i>statement_name</i>
STSN	Start New Session	none
STSV	Set Savepoint	dbname: <i>database_name</i> extra_1: <i>transaction_id</i> objname: <i>savepoint_name</i>
STTX	Set Transaction Mode	extra_1: <i>operation</i> ²⁰ flags: <i>mode_flags</i> ¹⁹ extra_2:
SVXD	Save External Directives	dbname: <i>database_name</i>

Table 16. Audit events listed by event code (continued)

Event Code	Event	Variable Contents
		objname: <i>active/inactive/test</i>
		objname: <i>directive_text</i>
TCTB	Truncate Table	dbname: <i>database_name</i>
		tabid: <i>table_id</i>
		objname: <i>table_name</i>
TMOP	Time Optical Cluster	flags: time flag ¹³
ULTB	Unlock Table	dbname: <i>database_name</i>
		tabid: <i>table_id</i>
UPAM	Update Audit Mask	login: user id
UPCK	Bring Chunk Online	extra_1: <i>chunk_number</i>
		flags: <i>mirror_status</i> ¹
UPDM	Enable Disk Mirroring	extra_1: <i>dbspace_number</i>
UPRW	Update Current® Row	dbname: <i>database_name</i>
		tabid: <i>table_id</i>
		extra_1: <i>old_part_number</i>
		row_num: <i>old_row_id</i> ¹⁴
		flags: <i>new_row_id</i>
		extra_2: <i>new_part_number</i>
USSP	Update Statistics, SPL Routine	dbname: <i>database_name</i>
		tabid: <i>proc_id</i>
USTB	Update Statistics, Table	dbname: <i>database_name</i>
		tabid: <i>table_id</i>

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 205](#) 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

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 214](#)

[The `onshowaudit` utility on page 221](#)

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 *HCL® 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

Use **ADTMODE** to control the level of auditing. The **ADTMODE** parameter is now deprecated; you should use the new **ADT_CLASSIC_ENABLED**, **ADT_DBSA**, and **ADT_DBSSO** parameters instead.

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)`

The `-l` option still exists but is obsolescent as it bundles three separate controls in one value. The new `-L`, `-S`, and `-A` options provide separate options for the three controls. The valid arguments for `-l` are still 0, 1, 3, 5, 7. The value 0 turns classic audit logging off (and also turns mandatory auditing for the DBSA and DBSSO off); the non-zero values all turn classic audit logging on. The value 1 turns classic audit logging on, without enabling mandatory auditing for DBSA or DBSSO users. The value 3 turns on mandatory DBSSO auditing but turns off mandatory DBSA auditing. Level 5 turns on mandatory DBSA auditing but turns off mandatory DBSSO auditing. Level 7 turns on mandatory auditing for both DBSA and DBSSO users. On the command line, the value for `-l` (ADTMODE) is processed before the values for `-L` (ADT_CLASSIC_ENABLED), `-S` (ADT_DBSSO), or `-A` (ADT_DBSA) are processed. The explicit options always override the values inferred from ADTMODE. The `-l` option corresponds to the ADTMODE parameter, but ADTMODE is no longer written to the configuration file. If the configuration file contains the ADTMODE parameter at start up, it will be recognized and acted on, but its settings will be overridden by appearances of ADT_CLASSIC_ENABLED, ADT_DBSSO or ADT_DBSA in the configuration file. The order in which these parameters appear is not material; the processing is as if the ADTMODE setting precedes the ADT_CLASSIC_ENABLED, ADT_DBSSO and ADT_DBSA settings.

The distributed `adtcfg` and `adtcfg.std` template files only contain ADT_ENABLED, ADT_DBSA, ADT_DBSSO settings, and only mention ADTMODE in comments.

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 *HCL® 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`)

ADT_CLASSIC_ENABLED configuration parameter

Use this configuration parameter to enable or disable classical (as opposed to syslog) auditing.

default value

0

range of values

0, OFF, FALSE, DISABLE, NO = Turn classic auditing off

1, ON, TRUE, ENABLE, YES = Turn classic auditing on

takes effect

When `onaudit` is run with `-L` option, it sets the `ADT_CLASSIC_ENABLED` parameter in the audit configuration file.

utility

`onaudit` (`onaudit -L on/off flag`)

This configuration parameter is preferred over `ADTMODE`.

ADT_DBSA configuration parameter

Use this configuration parameter to enable or disable the mandatory auditing for the DBSA group.

default value

0

range of values

0, OFF, FALSE, DISABLE, NO = Disable mandatory auditing for the DBSA group

1, ON, TRUE, ENABLE, YES = Enable mandatory auditing for the DBSA group

takes effect

When `onaudit` is run with `-A` option, it sets the `ADT_DBSA` parameter in the audit configuration file.

utility

`onaudit` (`onaudit -A on/off flag`)

ADT_DBSSO configuration parameter

Use this configuration parameter to enable or disable the mandatory auditing for the DBSSO group.

default value

0

range of values

0, OFF, FALSE, DISABLE, NO = Disable mandatory auditing for the DBSSO group

1, ON, TRUE, ENABLE, YES = Enable mandatory auditing for the DBSSO group

takes effect

When **onaudit** is run with -S option, it sets the ADT_DBSSO parameter in the audit configuration file.

utility

onaudit (**onaudit** -S *on/off flag*)

ADT_SYSLOG_ENABLED configuration parameter

Use this configuration parameter to enable or disable the Audit to Syslog (ASL) functionality.

default value

0

range of values

0, OFF, FALSE, DISABLE, NO = Turn ASL off

1, ON, TRUE, ENABLE, YES = Turn ASL on

takes effect

When **onaudit** is run with -E option, it sets the ADT_SYSLOG_ENABLED parameter in the audit configuration file.

utility

onaudit (**onaudit** -E *on/off flag*)

ADT_SYSLOG_FACILITY configuration parameter

Use this configuration parameter to help with filtering messages in the syslog configuration. The parameter is written in the configuration file and on the command line with commas separating separate options and no spaces.

default value

LOG_USER

range of values

LOG_USER (the default), LOG_LOCAL0..LOG_LOCAL7, LOG_AUTH, or LOG_AUTHPRIV, or LOG_AUDIT (if available).

takes effect

When **onaudit** is run with -F option, it sets the ADT_SYSLOG_FACILITY parameter in the audit configuration file.

utility

onaudit (**onaudit** -F *facility*)

The *facility* can be written with or without the LOG_ prefix and in upper-case or lower-case or mixed case. The LOG_ prefix and all upper-case is used when options are written to the audit configuration file.

ADT_SYSLOG_IDENTIFIER configuration parameter

Use this configuration parameter to choose the identifier name to be used in syslog messages.

default value

DBSERVERNAME from the ONCONFIG file

range of values

takes effect

When **onaudit** is run with -I option, it sets the ADT_SYSLOG_IDENTIFIER parameter in the audit configuration file. The maximum length allowed is 128 characters. The recommended maximum length is 32 characters.

utility

onaudit (**onaudit** -I *Identifier*)

ADT_SYSLOG_OPTIONS configuration parameter

Use this configuration parameter to specify options to openlog(). The parameter is written in the configuration file and on the command line with commas separating separate options and no spaces.

default value

LOG_NDELAY and LOG_NOWAIT

range of values

LOG_NDELAY, LOG_NOWAIT(the default)

LOG_NDELAY, LOG_ODELAY(mutually exclusive)

LOG_PERROR , LOG_CONS, LOG_PID

takes effect

When **onaudit** is run with -O option, it sets the ADT_SYSLOG_OPTIONS parameter in the audit configuration file.

utility

onaudit (**onaudit** -O *options*)

The *options* can be written with or without the LOG_ prefix and in upper-case or lower-case or mixed case. The LOG_ prefix and all upper-case is used when options are written to the audit configuration file.

ADT_SYSLOG_PRIORITY configuration parameter

Use this configuration parameter to specify a priority while filtering messages in the syslog daemon. The parameter is written in the configuration file and on the command line with commas separating separate options and no spaces.

default value

LOG_INFO

range of values

LOG_INFO(the default), LOG_EMERG, LOG_ALERT, LOG_CRIT, LOG_ERR, LOG_WARNING, LOG_NOTICE,
LOG_DEBUG

takes effect

When **onaudit** is run with -P option, it sets the ADT_SYSLOG_PRIORITY parameter in the audit configuration file.

utility

onaudit (**onaudit** -P *priority*)

Use of more urgent *priority* such as LOG_EMERG is not recommended.

The *priority* can be written with or without the LOG_ prefix and in upper-case or lower-case or mixed case. The LOG_ prefix and all upper-case is used when options are written to the audit configuration file.

Index

Special Characters

- _default mask 210
- _global mask 210
- _require mask 210
- 20361 SQLCODE 108
- 26456 SQLCODE 108
- 30020 SQLCODE 108
- 30090 SQLCODE 108
- 32509 SQLCODE 108
- 387 SQLCODE 108
- \$INFORMIXDIR
 - checking security 7
 - disabling security check of directory and subdirectories 13
 - fixing security problem 7
 - onconfig file 4
 - permissions 4, 5
 - Permissions
 - on \$INFORMIXDIR and subdirectories 13
 - Permissions, UNIX 13
 - security of the installation path 4
 - sqlhosts
 - permissions 4
 - subdirectories 5
 - trusted group 7
 - trusted user 7
 - trusted.insecure.directories file 7

A

- aaodir directory 6, 186
- Access control
 - authentication 122
- Access privileges
 - middleware servers 100
 - three-tier application model 100
 - trusted connections 100
- Access privileges, Windows 180, 185, 186
- Access to audit trail, controlling 176, 177, 178, 205
- account function 112
- Adding audit masks 194
- ADDRESS attribute
 - trusted connections 101, 102
 - trusted context objects 101, 102
- Administrative roles
 - audit analysis officer 186
 - database administrator 186
 - database server administrator 185
 - database system security officer 185
 - listed 168
 - operating-system administrator 186
- Administrator
 - audit analysis officer 186
 - database 186
 - database server 185
 - database system security officer 185
 - operating system 186
- ADT_CLASSIC_ENABLED 255
- ADT_DBSA 255
- ADT_DBSSO 255
- ADT_SYSLOG_ENABLED 256
- ADT_SYSLOG_FACILITY 256
- ADT_SYSLOG_IDENTIFIER 257
- ADT_SYSLOG_OPTIONS 257
- ADT_SYSLOG_PRIORITY 257
- ADTCFG file
 - aaodir directory 186
 - audit configuration
 - UNIX 173
 - Windows 173
 - configuration parameters 199, 199
 - conventions used 250, 251
 - description of 251
 - white space 251
- ADTERR configuration parameter 199, 199, 252
- adtmasks.std file 196
- ADTMODE 255
- ADTMODE configuration parameter 199, 199, 252
- ADTPATH configuration parameter 199, 199, 253
- ADTROWS configuration parameter 254
- ADTSIZE configuration parameter 199, 199, 254
- Advanced Encryption Standard 16, 18
- AES. 16
- Aggregation 180
- ALTER SECURITY LABEL COMPONENT statement 144
- ALTER TABLE statement 148, 150, 150
- ALTER TRUSTED CONTEXT statement
 - trusted connections 101
- ALTER USER statement 94
- Application Event log, Windows 176
- archecker utility 155
- ARRAY
 - see security label component 141
- Attributes
 - ADDRESS 101, 102
 - DEFAULT ROLE 101, 104
 - DISABLE 101
 - ENABLE 101
 - NO DEFAULT ROLE 101
 - PUBLIC 101, 103
 - ROLE 101, 104, 105
 - SQL_ATTR_USE_TRUSTED_CONTEXT 105
 - TCTX=1 105
 - WITH AUTHENTICATION 101, 103
 - WITH USE FOR 101, 103
 - WITHOUT AUTHENTICATION 101, 103
- Audit
 - features 164
 - performance 171
 - process for 167
 - reasons for 164
 - record format 203
 - turning on auditing 194
- Audit administrator
 - audit analysis officer 168, 185
 - audit configuration 167, 173
 - audit instructions 170
 - audit masks 165, 169
 - audit-trail analysis 164
 - auditing on or off 169, 173
 - database system security officer 168, 185
 - roles 168, 185
 - security risk 169
- Audit analysis
 - creating a data file 206
 - importance of 179
 - loading audit data into a database 209
 - overview 179
 - preparing for 179, 179
 - records indicating event failure 180
 - records indicating event success 180
 - strategies for 180
 - with SQL
 - creating a command file 208
 - creating a database 207
 - creating a table 207
 - description 205
 - performing 205
 - preparing for 206
 - without database 204
 - without SQL 204, 204
- Audit analysis officer (AAO)
 - audit administrator 168, 185
 - role description 186
 - security threats 182
 - UNIX 186
 - Windows registry settings 186
- audit configuration 214
 - showing
 - from a command line 214
 - with onshowaudit 214
- Audit configuration
 - ADTCFG file 173
 - showing
 - from a command line 199
- Audit data
 - controlling access to 205
 - creating a table for 207
 - loading into database 209
 - privileges to protect 205
- Audit error mode
 - changing 202
 - in ADTCFG file 252
 - setting 191
- Audit events
 - alphabetical listing of codes 225
 - displaying 197
 - fields shown 225
 - listed 225
- Audit files, UNIX
 - controlling access to 176
 - directory
 - specifying with ADTPATH 253
 - error modes when writing to 176
 - specifying maximum size
 - with ADTSIZE 254
- Audit instructions
 - resource and performance implications 171
 - who sets 170
- Audit level, setting 191
- audit masks
 - creating from a command line 210
 - deleting 210
 - modifying
 - command syntax for 210
 - showing 210
- Audit masks
 - _default mask 169
 - _exclude mask 169
 - _require mask 169
 - adding 195
 - base mask 195
 - compulsory masks 169
 - conflict in audit instructions 169
 - creating a template 195
 - creating a user mask from a template mask 195
 - deleting 198

- displaying 197
- how to use 172
- individual user mask 169
- maintaining 194
- modifying
 - from a command line 198
 - from an input file 196
 - instructions 198
- restricted names 170
- setting up default and compulsory 190
- templates 170
- types, listed 168
- user mask 169

Audit records

- controlling access to 176
- interpreting extracted information 209

audit trail

- starting auditing from a command line 214

Audit trail

- administration 194, 197
- controlling access to 176, 178
- operating-system, UNIX 167
- reviewing 168
- starting a new UNIX file 200
- UNIX file permissions 177, 177, 177, 177, 177, 178
- UNIX files 176
- Windows access privileges 178
- Windows Application Event log 176

Audit trail, controlling access to 178

auditing

- error mode levels 214
- turning off 214
- turning on 214

Auditing

- ADTCFG file
 - UNIX 173
 - Windows 173
- creating user masks from template masks 195
- displaying fragmentation information 172
- granularity 172
- IBM
 - Security
 - Guardium
 - 159
 - selective row-level 192
 - setting the level 191
 - setting up 189
 - specifying UNIX directory
 - with ADTPATH 253
 - turning off 173, 202
 - turning on 173, 194
- Auditing user access
 - middleware servers 100
- auth function 112
- authentication 86, 86
- Authentication
 - modules 84, 122
 - single sign-on 84, 122, 122
 - and Kerberos layer 124
 - and keytab file 127, 128
 - trusted connections 103, 203
 - trusted context objects 103
 - types 122

B

- backup and restore 155
- Base mask, defined 195
- Blowfish 16
- Browsing 180

C

- cfid option 91
- Changing the audit error mode 202
- chunk files 15
- Cipher
 - AES 18, 21
 - Blowfish 18
 - defined 16
 - DES 18, 21
 - in encryption 18
 - supported types 18, 18, 21
- cipher encryption tag 22
- ciphers
 - switch frequency 21
- Client software
 - single sign-on 124
- Clients
 - single sign-on 129, 130
- column level data protection 148, 150
- column-level data protection 136
- Command files
 - creating for dbload 208
 - use with dbload 208
- communication files directory option 91
- Communication Support Module 16, 119
 - concsm.cfg entry 22, 121
 - configuration file 17, 17, 119, 119
 - for single sign-on (GSSCSM) 122
- Compulsory audit masks
 - setting up 190
 - when applied 169
- concsm.cfg file 16, 17, 119, 119
 - building SMI tables 121
 - entry for network data encryption 22, 22
 - entry for password encryption 121
 - entry for single sign-on 126
 - location 17, 119
 - single sign-on
 - configuration 126
- confidentiality service 122, 126
- configuration parameters
 - GSKIT_VERSION 29, 37
 - PLCY_HASHSIZE 152
 - PLCY_POOLSIZ 152
 - USRC_HASHSIZE 152
 - USRC_POOLSIZ 152
- Configuration parameters
 - ADTERR 199, 199, 252
 - ADTMODE 199, 199, 252, 252
 - ADTPATH 199, 199, 253
 - ADTROWS 254
 - ADTSIZE 199, 199, 254
 - CREATE USER statement 93
 - DB_LIBRARY_PATH 16
 - DBCREATE_PERMISSION 134
 - described 250
 - IFX_EXTEND_ROLE 134
 - listed 199, 199
 - LISTEN_TIMEOUT 131
 - MAX_INCOMPLETE_CONNECTIONS 131
 - SECURITY_LOCALCONNECTION 130
 - UNSECURE_ONSTAT 135
 - USERMAPPING 93
- Configuration, audit
 - displaying 199, 199
 - maintaining 199
 - overview 173
 - tasks listed 173
- Configuring role separation 188
- CONNECT statement

- TRUSTED keyword 105
- Connection security
 - middleware servers 100
 - three-tier application model 100
- Continue error modes 176
- Controlling access to audit trail 176, 176, 177, 178, 205
- Coserver 253
- CREATE ROLE statement 132
- CREATE SECURITY LABEL COMPONENT statement 144
- CREATE SECURITY LABEL statement 146
- CREATE SECURITY POLICY statement 145
- CREATE TABLE statement 148, 150, 150
- CREATE TRUSTED CONTEXT statement
 - trusted connections 101
- Creating a data file 206
- Creating a database for audit data 207
- Creating a table for audit data 207
- Creating a user mask from a template mask 195
- creating an audit mask from a command line 210
- Credentials 122
 - single sign-on 127, 128, 128

D

Data

- audit, loading into database 209
- creating a file for dbload 206
- extracting with onshowaudit 204
- transmission encryption 16

data classifications 139

data definition language (DDL)

- with label-based access control (LBAC) 155

Data encryption 16, 16

Data Encryption Standard 16, 16

data loading and unloading 155

Data Server Driver for JDBC and SQLJ

- trusted connection requests 100

Data Server Provider for .NET

- trusted connection requests 100

Database administrator (DBA) 186

Database connections

- middleware servers 100
- three-tier application model 100

database security administrator

- see DBSECADM 136

database server administrator (DBSA) 139

Database server administrator (DBSA)

- administrative role 185
- in label-based access control 136, 139
- role description 185
- security threats 182

database servers

- managing auditing
 - with onaudit 214

Database servers

- groups 14
- managing auditing
 - with ADTMODE 252
- monitoring events and users 191
- Password Communication Support Module 121
- quiescent mode 173

Database system security officer (DBSSO)

- audit administrator 168, 185
- role description 185
- security threats 182
- UNIX 185
- Windows registry settings 185

- Databases
 - creating for
 - HCL
 - Informix
 - audit records
 - 207
 - sysmaster 199
 - DataBlade
 - restricting access for registering UDRs 134
 - DB_LIBRARY_PATH configuration
 - parameter 16
 - DBCREATE_PERMISSION configuration
 - parameter 134
 - dbexport utility 155
 - dbimport utility 155
 - dbload utility 155
 - creating a command file for 208
 - creating a data file for 206
 - creating a database for 207
 - creating a table for 207
 - loading audit data into a database 209
 - DBMS security threats 182
 - dbschema utility 155
 - DBSECADM 136, 138, 139, 139, 145, 146, 146, 147, 151, 152, 153
 - DBSERVERALIASES
 - single sign-on 125
 - dbservername
 - single sign-on 125
 - dbssodir directory 6, 185
 - debug files 91
 - Default audit mask 169
 - setting up 190
 - when applied 169
 - DEFAULT_ROLE attribute
 - trusted connections 101, 104
 - trusted context objects 101
 - trusted contexts 104
 - DEFAULT_ROLE keywords
 - trusted connections 104
 - trusted contexts 104
 - default roles
 - assigning 104
 - in trusted-context objects 104
 - inheriting 104
 - Defaults
 - role
 - creating 133
 - granting privileges 133
 - using 133
 - Deleting audit masks 198, 210
 - Denial-of-service flood attacks 5, 131
 - DES. 16
 - DES3. 16
 - digital certificates 29, 37
 - Directories
 - aaodir 186
 - securing 3
 - specifying for UNIX audit files
 - with ADTPATH 253
 - with onaudit 190
 - DISABLE attribute
 - trusted connections 101
 - trusted context objects 101
 - discretionary access control 132
 - Discretionary Access Control (DAC) 182
 - displaying
 - audit masks 210
 - Displaying
 - audit configuration 199, 199
 - audit masks 197

- Distributed database configuration threats 184
- distributed queries 91, 158
- DROP SECURITY LABEL COMPONENT 153
- DROP SECURITY LABEL statement 153
- DROP SECURITY POLICY statement 153
- DROP TRUSTED CONTEXT statement
 - trusted connections 101
- DROP USER statement 94
- dropping
 - security label components 153
 - security labels 153
 - security policies 153
- Dynamic Host Configuration Protocol (DHCP)
 - trusted connections 101

E

- elements
 - for label-based access control 137
- ENABLE attribute
 - trusted connections 101
 - trusted context objects 101
- Enable Role Separation check box 188
- ENCCSM
 - Communication Support Modules, encryption 16
- ENCCSM_CIPHERS encryption parameter 25, 26
- ENCCSM_MAC encryption parameter 25, 26
- ENCCSM_MACFILES encryption parameter 25, 27
- ENCCSM_SWITCH encryption parameter 25, 27
- ENCRYPT function 80
- Encrypting
 - column data 82
- encryption
 - switch frequency 21
- Encryption
 - column storage consideration 80
 - column-level 80, 82
 - credentials in SSO 126
 - data transmissions 16, 119
 - defined 16
 - enabling with communication support modules 17
 - example
 - encrypting a column 84
 - querying encrypted data 84
 - size of an encrypted column 83
 - modes 16
 - of data in a column 80, 82
 - of network data transmissions 16
 - of passwords 16
 - password 119
 - passwords 119
 - single sign-on 122, 122
 - Users
 - user IDs 122
 - encryption parameter
 - ENCCSM_CIPHERS 26
 - encryption parameters
 - example 28
 - overview 25
 - encryption tags 22
 - example 22
 - mac tag 22
 - switch tag 22
- Enforcing role separation 188
- Enterprise Replication 155
- Environment variables
 - INF_ROLE_SEP 188

- INFORMIXCONCSMCFG 17, 119
- INFORMIXDIR 4
- NODEFDAC 205
- Error messages
 - for server utilities security check 13
 - trusted connection switching 108
- Error messages log, size of 176
- Error mode
 - and ADTERR 252
 - changing 202
 - continue 176
 - halt 176
 - implications of 191
 - setting 191
 - when writing to an audit file 176
- ESQL/C API
 - trusted connection requests 100
- event alarm
 - in audit trail 167
- Event codes, alphabetical listing 225
- Event failure 180
- Event success 180
- Events
 - codes listed 225
 - defined 165
 - fields shown 225
 - level of auditing for specified 172
- Examples
 - trusted connection request 105
 - trusted connection switch request 107
 - trusted context object 102, 103, 104, 105
- Exclude audit mask 169
- exemptions 151, 151, 152
- explain files 91
- External modules
 - security for loading 16
- External routines
 - security for 134

F

- Fields for audit events 225
- FILE statement 208
- Files
 - data, creating for dbload 206
 - input
 - for modifying masks 196
 - UNIX audit
 - controlling access to 176, 177
- Format
 - for audit records 203
 - for dbload data file 207
- fragmentation 158
- Fragmentation, information in audit events 172
- functions
 - security label support 148

G

- generated files 91
- Generic Security Services
 - API 122
 - communications support module 122
- getDB2TrustedPooledConnection method
 - trusted connection requests 105
 - trusted connection switching 107
- getDB2TrustedXAConnection method
 - trusted connection requests 105
 - trusted connection switching 107
- Global Security Kit 29, 37
- GRANT DBSECADM statement 139
- GRANT EXEMPTION statement 151
- GRANT SECURITY LABEL statement 146

- GRANT statement 132
 - when granting privileges to DataBlade users 134
- Group
 - database server 14
 - trusted 5
 - trusted group 5
- Group informix 14
 - installation path 5
- GSKCapiCmd 29, 37
- GSKCmd 29, 37
- GSKikm 29, 37
- GSKit 37
- GSKIT_VERSION configuration parameter 29, 37
- Guest account on windows 99
- Guidelines for assigning roles 187

H

- Halt modes 176
- HCL
 - Informix
 - audit record format for 203
 - extracting and loading audit records for 206
- HCL
 - Informix
 - ESQL/C API
 - switching users on a trusted connection 107
 - trusted connection requests 100, 105
- HCL
 - Informix
 - JDBC Driver
 - switching users on a trusted connection 107
 - trusted connection requests 100, 105
- HCL
 - Informix
 - ODBC Driver
 - switching users on a trusted connection 107
 - trusted connection requests 100, 105
- hierarchical tables 136
- high-availability solutions 158
- High-Performance Loader 155
- home directory 91

I

- IBM Data Server Driver for JDBC and SQLJ
 - switching users on a trusted connection 107
 - trusted connection requests 100, 105
- IBM Data Server Provider for .NET
 - switching users on a trusted connection 107
 - trusted connection requests 100, 105
- IBM
 - Security
 - Guardium
 - 159
 - IDSLBACREADARRAY rule 141
 - IDSLBACREADSET rule 142
 - IDSLBACREADTREE rule 142
 - IDSLBACWRITEARRAY rule 141
 - IDSLBACWRITESET rule 142
 - IDSLBACWRITETREE rule 142
 - IDSSecurityLABEL data type 148
 - IFX_EXTEND_ROLE configuration parameter 134
 - ifxguard configuration file 159
 - IFXGUARD configuration parameter 162
 - ifxguard utility 160

- IKEYCMD 29, 37
- iKeyman 29, 37
- INF_ROLE_SEP environment variable 188
- informix user account 176, 186, 186
- INFORMIXCONCSMCFG environment variable 17, 119
- INFORMIXDIR environment variable 4
- Insider attack 180
- installation directory
 - see \$INFORMIXDIR
- Integrity service 122, 126
- internal authentication 87
- internal users 87
- Internal users 86
- ipload utility 155
- IPv4 addresses
 - trusted connections 101, 102
 - trusted context objects 101, 102
 - trusted locations 101, 102
- IPv6 addresses
 - trusted connections 101, 102
 - trusted context objects 101, 102
 - trusted locations 101, 102
- istar 91
- IXUSERS seccfg setting 188

J

- JDBC Driver
 - trusted connection requests 100

K

- Kerberos 122, 124
 - authentication
 - single sign-on 128
 - testing setup for SSO 128
- Key Distribution Center 122
- keystores 29, 37
- keytab file 124, 127, 128
- Keywords
 - ADDRESS 101, 102
 - ATTRIBUTES 101
 - BASED UPON CONNECTION USING SYSTEM AUTHID 101
 - DEFAULT ROLE 101
 - DISABLE 101
 - ENABLE 101
 - NO DEFAULT ROLE 101
 - PUBLIC 101, 103
 - ROLE 101
 - USER 101, 103
 - WITH AUTHENTICATION 101, 103
 - WITH USE FOR 101, 103
 - WITHOUT AUTHENTICATION 101, 103

L

- label-based access control
 - and other
 - Informix
 - features
 - 158
 - configuration parameters 152
 - configuring 136
 - maintaining 152
 - planning 139
 - read and write access 137
 - row and column protection on one table 148
 - security precautions 155
- LDAP Authentication Support
 - configuration file 113
 - installing 113
- LDAP Authentication Support on Windows 113

- LDAP module 86
 - application development 115
 - authentication 114
 - client APIs 118
 - compatibility issues 119
 - configuring 114
 - configuring
 - HCL
 - Informix
 - 114
 - distributed transactions 117
 - implicit connections 115
- LDAP server 113
- Level of auditing, determining 191
- LISTEN_TIMEOUT configuration parameter 131, 131
- Listener threads 131
- Loading onshowaudit data into a database table 209
- LOG_ALERT 257
- LOG_AUTH 256
- LOG_AUTHPRIV 256
- LOG_CRIT 257
- LOG_DEBUG 257
- LOG_EMERG 257
- LOG_ERR 257
- LOG_INFO 257
- LOG_NDELAY 257
- LOG_NOTICE 257
- LOG_NOWAIT 257
- LOG_USER 256
- LOG_WARNING 257

M

- mac encryption tag 22
- MAC key files
 - generating new 20
 - generation levels 21
 - overview 20
- Malicious software security threats 183
- Malware
 - see Malicious software security threats 183
- mapped user 91
- mapped users 87
- mask
 - deleting 210
 - modifying 210
 - with onaudit 210
 - showing with onaudit 210
- Mask
 - _default 169
 - _exclude 169
 - _require 169
 - creating
 - template 195
 - user mask from a template mask 195
 - user mask without a template mask 196
 - with onaudit 210
 - deleting 198
 - displaying 197
 - how to use 172
 - modifying
 - from an input file 196
 - from the command line 198
 - setting up compulsory 190
 - setting up default 190
 - template 170
 - types, listed 168
 - user 169
- MAX_INCOMPLETE_CONNECTIONS configuration parameter 131, 131

- Message Server service 176
- Methods
 - getDB2TrustedPooledConnection 105, 107
 - getDB2TrustedXAConnection 105, 107
- Middleware servers
 - connection security 100
 - database connections 100
 - overview 100
- Modifying audit masks 198
- modules 86
- Mutual authentication 122

N

- Named pipes interprocess communications 176
- Network
 - data encryption 22
- Network data transmissions encryption of 16
- NO DEFAULT ROLE attribute
 - trusted connections 101
 - trusted context objects 101
- NODEFDAC environment variable 205
- non-root installation 7
 - connections 91

O

- Obsolete user security threats 184
- ODBC Driver
 - trusted connection requests 100
- onaudit utility
 - ADTERR parameter 252
 - ADTMODE parameter 252
 - ADTPATH parameter 253
 - ADTROWS parameter 254
 - ADTSIZE parameter 254
 - audit configuration 214
 - audit events, adding to audit masks 190
 - audit masks
 - creating 210
 - deleting 198, 210
 - described 170
 - displaying 197
 - showing from command line 210
 - auditing on or off 173
 - changing the audit error mode 202
 - description of 210
 - displaying the audit configuration 199
 - error modes 176
 - error-mode levels 214
 - fragmentation information 172
 - HDR limitations 167
 - level of auditing for certain events 172
 - masks, modifying 196, 210
 - options 210
 - overview 210
 - setting the error mode 191
 - showing the audit configuration 214
 - specifying a directory for UNIX audit files 190
 - starting a new UNIX audit file 214
 - storage of audit records 214
 - template mask
 - creating 195
 - creating a user mask from 195
 - creating a user mask without 196
 - turning off auditing 202
 - turning on auditing 194
 - UNIX operations 210
 - used by AAO 186
 - used by DBSSO 185
 - who can run 210

- Windows operations 210
- onbar utility 155, 155
- oncheck utility 155
- onconfig file 173
- Online mode 173
- onload utility 155
- onlog utility 155
- onpload utility 155
- onsecurity utility 4, 7
 - Permissions
 - on chunk files 15
 - using with chunk files 15
- onshowaudit utility
 - audit trail access 176
 - data extraction from audit trail 168, 179
 - extracting data for audit analysis 204
 - listing of audit events for analysis 225
 - output accessible by AAO 182
 - role separation 176
 - used by AAO 186
 - using dbload with 207, 207
- ontape utility 155
- onunload utility 155
- Operating system
 - coordinating auditing between AAO and OSA 186
 - protected subsystem for audit trail 179
- Operating-system administrator (OSA)
 - administrative role 186
 - role defined 186
 - security threats 182
- Operating-system audit trail, UNIX 167
- options field
 - cfid option 91
 - communication files directory option 91

P

- PAM service
 - defining 112
- Parameters, configuration
 - ADTERR 199, 199, 252
 - ADTMODE 199, 199, 252
 - ADTPATH 199, 199, 253
 - ADTROWS 254
 - ADTSIZE 199, 199, 254
 - described 250
 - listed 199, 199
- Password Communication Support Module 121
- Password encryption
 - CSM configuration file 17, 119, 121
 - database server initialization 120
- Password storage
 - middleware servers 100
 - three-tier application model 100
- Path, specifying for auditing
 - with ADTPATH 253
- Performance implications of auditing 171
- Performing SQL audit analysis 205
- Permissions
 - for creating databases 134
 - installation path and subdirectories 4
- Permissions, UNIX 180, 185, 186
- Pluggable Authentication Module 86, 110
 - application development 115
 - authentication mode 111
 - client APIs 118
 - compatibility issues 119
 - configuring the connection 112
 - defined 110
 - distributed transactions 117

- implicit connections 115
 - required stack size 111
 - service name 111
 - supported platforms 110
- Preparing for audit analysis 179, 206
- Primary security threats 182
- Principals 122, 124
 - validating 128
- privacy policy
 - label-based access control 139
- Privileged activity security threats 182
- Privileged environment, security threat from untrusted software 184
- Privileged users 186
- Privileges to protect audit data 205
- Properties
 - TRUSTED_CONTEXT=TRUE 105
 - TrustedContextSystemPassword 105, 107
 - TrustedContextSystemUserID 105, 107
- protected data 145, 148
- PUBLIC attribute
 - trusted connections 101
 - trusted context objects 101
 - public directory permissions 6

Q

- Queries by browsers 180
- Quiescent mode 173

R

- race condition 5
- Raw audit records 179
- read access
 - installation path 6
 - label-based access control 137
- referential integrity scans 158
- Registry settings, Windows
 - for AAO 186
 - for DBSSO 185
 - for role separation 188
- Remote access to data, security threat 184
- REMOTE_SERVER_CFG 86
- REMOTE_USERS_CFG 86
- RENAME SECURITY LABEL COMPONENT statement 154
- RENAME SECURITY LABEL statement 154
- RENAME SECURITY POLICY statement 154
- RENAME TRUSTED CONTEXT statement
 - trusted connections 101
- RENAME USER statement 94
- renaming
 - security objects 154
- Require audit mask 169
- Resource implications of auditing 171
- Responding to security problems 181
- REVOKE DBSECADM statement 139
- REVOKE DEFAULT ROLE statement 133
- REVOKE EXEMPTION statement 152
- REVOKE SECURITY LABEL statement 147
- REVOKE statement
 - when granting privileges to DataBlade users 134
- Role
 - creating 133
 - default 133
 - defined 133
 - GRANT DEFAULT ROLE statement 133
 - overview 132
 - separation 133
- ROLE attribute
 - trusted connections 104, 105
 - trusted context 101

- trusted contexts 104, 105
- ROLE keyword
 - trusted connections 104, 105
 - trusted contexts 104, 105
- Role separation and onshowaudit 176
- Role Separation dialog box 185, 188
- roles
 - assigning 105
 - in trusted-context objects 105
 - inheriting 105
- Roles
 - administrative, listed 168
 - assigning 187
 - audit analysis officer 186
 - configuring and enforcing 188
 - database administrator 186
 - database server administrator 185
 - database system security officer 185
 - no separation, security configuration for 177
 - operating-system administrator 186
 - separation 177, 187, 188, 188
- root user account 186
- row and column protection on one table 148
- row level data protection 148, 150
- row-level data protection 136

S

- seccfg file 188
- SECLABEL functions 145
 - see security label support functions 148
- Secure domain names
 - trusted connections 101, 102
 - trusted context objects 101, 102
 - trusted locations 101, 102
- Secured Socket Layer protocol
 - overview 29
- Security
 - disabling server utilities check 13
 - encryption options 16
 - for DataBlade user-defined routines 134
 - for external routines 134
 - for loading external modules 16
 - middleware servers 100
 - Pluggable Authentication Module 110
 - preventing denial-of-service flood attacks 131
 - resetting directory permissions 12
 - server utilities check before starting on UNIX 4
 - three-tier application model 100
 - through LDAP Authentication Support 113
 - through roles 133
 - using column-level encryption 80
 - using Communication Support Modules 16, 119
- Security configuration for audit files 177
- security label component
 - ARRAY 141
 - SET 142
- security label components 139
 - altering 144
 - creating 144
 - in exemptions 151
 - TREE 142
- security label support functions 148
- security labels 145
 - creating 146
 - functioning 137
 - granting 146
 - revoking 147

- security policies 145
 - creating 145
- Security threats
 - aggregation 180
 - audit analysis officer 182
 - browsing 180
 - database server administrator 182
 - database system security officer 182
 - DBMS 182
 - distributed databases configuration 184
 - granting remote access to data 184
 - insider attack 180
 - malicious software 183
 - obsolete user 184
 - operating-system administrator 182
 - primary 182
 - privileged activity 182
 - responses to 181
 - setting the auditing level 191
 - shared-memory connection 183
 - untrusted software in privileged environment 184
- SECURITY_LOCALCONNECTION configuration parameter 130
- selective row-level auditing 192, 214, 254
- SERVERNUM configuration parameter 173
- Session, effects of errors 176
- SET
 - see security label component 142
- SET ENCRYPTION PASSWORD statement 80
- set group ID (SGID) 14
- SET ROLE DEFAULT statement 133, 133
- SET SESSION AUTHORIZATION statement 155
 - trusted connection switching 107
 - trusted connections 107
- SET statement 132
- set user ID (SUID) 14
- setenv utility 188
- Shared-memory connection 183
- showing
 - audit masks 210
- Simple Password Communication Support Module
 - CSM configuration file 121
- single sign-on 126
- Single sign-on 122, 123
 - and Kerberos protocol 122
 - clients with 129, 130
 - configuring the database server 125
- Size, specifying maximum for UNIX audit files
 - with ADTSIZE 254
- SMI sysadinfo table 199
- SMI tables
 - concsm.cfg 121
- SPWDCSM 16, 119, 120
- SQL statement
 - SET SESSION AUTHORIZATION 107
- SQL statements
 - ALTER TRUSTED CONTEXT 101
 - CONNECT 105
 - CREATE DATABASE 207
 - CREATE ROLE 132
 - CREATE TRUSTED CONTEXT 101
 - DROP TRUSTED CONTEXT 101
 - GRANT 132, 205
 - GRANT DEFAULT ROLE 133
 - RENAME TRUSTED CONTEXT 101
 - REVOKE 205
 - REVOKE DEFAULT ROLE 133
 - SET ROLE 132
 - SET ROLE DEFAULT 133, 133

- SQLSetConnectAttr function
 - trusted context 105
- SQL_ATTR_USE_TRUSTED_CONTEXT attribute
 - trusted context 105
- SQLCODE values
 - 26456 108
 - 30020 108
 - 30090 108
 - 32509 108
 - 387 108
- sqlhosts
 - for single sign-on 125
 - path name for UNIX 183
- sqlhosts file
 - cfid option 91
 - communication files directory option 91
- SSL protocol
 - overview 29
- SSL_KEYSTORE_LABEL configuration parameter 29
- Stored procedures
 - trusted connection switching 108
- Strategies for audit analysis 180
- Superuser (root) 177
- surrogate user properties 87
- switch encryption tag 22
- switch frequency
 - encryption 21
- Switching User IDs
 - rules 108
 - trusted connections 108
- Switching users
 - trusted connections 103, 107
 - trusted context objects 103
- synonyms 158
- Syntax
 - onsecurity utility 7
- sysadinfo table 199
- sysaudit table 170
- sysintausers tables 96
- sysmaster database 170
- sysmaster database, sysadinfo table 199
- sysroleauth table 134
- sysurrogategroups table 96
- sysurrogateusers table 96
- System performance
 - middleware servers 100
 - three-tier application model 100
 - trusted connections 100
- sysuser database 96, 117
- sysusermap table 96

T

- Table
 - creating for audit data 207
 - sysadinfo 199
- TCP/IP
 - trusted connections 100
- Template audit masks 170
 - base mask 195
 - creating from user masks 195
 - creating with onaudit 195
 - description 170
 - naming rules 170
- temporary (TEMP) tables 136, 155
- Threads, suspended 176
- Three-tier application model
 - connection security 100
 - overview 100
- Tickets
 - Kerberos authentication 122

- TLS 29
- Transaction boundaries
 - trusted connection switching 108
- Transport Layer Security 29
- Triple Data Encryption Standard 16
- Trusted connection requests
 - HCL
 - Informix
 - ESQL/C API
 - 105
 - HCL
 - Informix
 - JDBC Driver
 - 105
 - HCL
 - Informix
 - ODBC Driver
 - 105
 - IBM Data Server Driver for JDBC and SQLJ 105
 - IBM Data Server Provider for .NET 105
- Trusted connection switching
 - HCL
 - Informix
 - ESQL/C API
 - 107
 - HCL
 - Informix
 - JDBC Driver
 - 107
 - HCL
 - Informix
 - ODBC Driver
 - 107
 - IBM Data Server Driver for JDBC and SQLJ 107
 - IBM Data Server Provider for .NET 107
 - Trusted connections
 - APIs 100
 - assigning a default role 104
 - assigning a role 105
 - authentication 103
 - communication protocols 100
 - creating 105
 - default user roles 104
 - inheriting a default role 104
 - inheriting a role 105
 - requesting 105
 - requirements 100
 - switching 107
 - switching rules 108
 - switching users 103
 - user authentication 103
 - user roles 105
 - Trusted context objects
 - attributes 101
 - authentication 103
 - creating 101
 - keywords 101
 - requirements 100
 - switching users 103
 - user authentication 103
 - TRUSTED keyword
 - CONNECT statement 105
 - Trusted locations
 - Dynamic Host Configuration Protocol (DHCP) 101
 - trusted connections 101, 102
 - trusted context objects 101, 102
 - trusted user 5
 - TRUSTED_CONTEXT=TRUE property 105

- Trusted-context objects
 - assigning a default role 104
 - assigning a role 105
 - default user roles 104
 - inheriting a default role 104
 - inheriting a role 105
 - user roles 105
- TrustedContextSystemPassword property
 - trusted connections 105, 107
- TrustedContextSystemUserID property
 - trusted connections 105, 107
- typed tables 136
- types 86

U

- UNIX
 - ADTCFG file 173
 - audit configuration 173
 - audit files
 - directory 190, 253
 - new 200
 - size 254
 - audit-trail files 176
 - machine notes file 179
 - operating-system audit trail 167
 - operations with onaudit 210
 - permissions 185, 186
 - workstations 184
- Unscrupulous user 169, 180, 185, 187
- UNSECURE_ONSTAT configuration parameter 135
- Untrusted software 184
- User authentication
 - middleware servers 100
 - three-tier application model 100
 - trusted connections 103
 - trusted context objects 103
- User authorization
 - middleware servers 100
 - three-tier application model 100
- User ID storage
 - middleware servers 100
 - three-tier application model 100
- user informix
 - running onaudit 210
- User informix
 - audit files owner 177
 - installation path 5
 - retrieving audit configuration information 199
 - running onshowaudit 176
- User mapping tables 96
- User mask
 - and _default mask 169
 - creating from a template mask 195
 - creating without a template mask 196
- user operations
 - in label-based access control 137
- user permissions 5
- user roles
 - default roles 133
 - overview 132
 - role separation 133
- user-defined routines 155
- User-defined routines
 - external 134
 - registering 134
- USERMAPPING configuration parameter 93
- Users
 - accounts with same name 184
 - auditing 169

- privileged 186
- system 186
- trusted 5
- user IDs 124
- utilities
 - onaudit 210
- Utilities
 - onsecurity 4, 7
 - setenv 188
- utility
 - onaudit utility 210

V

- values
 - for label-based access control 137
- views 158
- violations tables 158
- virtual-index interface (VII) tables 136
- virtual-table interface (VTI) tables 136

W

- Warning messages
 - for server utilities security check 13
- White space in ADTCFG file 251
- Windows
 - access privileges 185, 186
 - access privileges for audit trail 178, 180
 - ADTCFG file 173
 - Application Event log 176, 176
 - audit configuration 173
 - audit trail in Application Event log 176
 - Guest account 99
 - operations with onaudit 210
 - registry settings
 - for AAO 186
 - for DBSSO 185
 - for role separation 188
- WITH AUTHENTICATION attribute
 - trusted connections 101
 - trusted context objects 101
- WITH USE FOR attribute
 - trusted connections 101
 - trusted context objects 101
- WITHOUT AUTHENTICATION attribute
 - trusted connections 101
 - trusted context objects 101
- working directory 91
- write access
 - installation path 6
 - label-based access control 137

Z

- Zero (0) 199
 - ADTERR setting 199
 - ADTMODE default value 252
 - continue error code 176
 - onaudit error mode 191